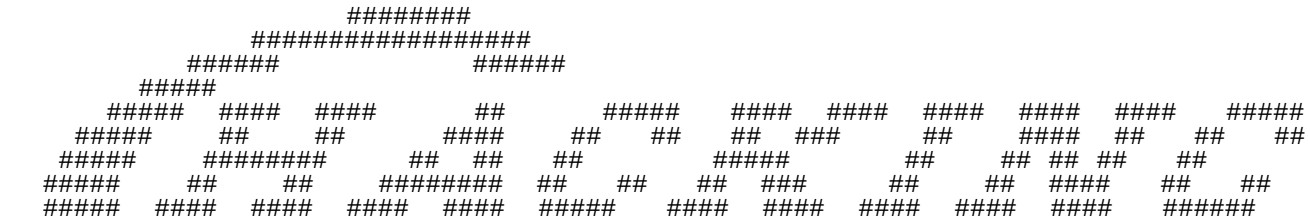


Issue #18

July 3, 1999

.....

Time flies like the wind; fruit flies like a banana.

"Yet if the only form of tradition, of handing down, consisted in following the ways of the immediate generation before us in a blind or timid adherence to its successes, 'tradition' should be positively discouraged. We have seen many such simple currents lost in the sand; and novelty is better than repetition. [Tradition] cannot be inherited, and if you want it you must obtain it by great labor."

T. S. Eliot, "Tradition and the Individual Talent"

.....

BSOUT

Tradition. C=Hacking is a magazine of considerable tradition, a tradition that is not merely imitating the successes of previous generations. C=Hacking has a tradition of true excellence, and once again, great people have contributed their time and talents towards the 64 community, and another fine issue of C=Hacking. And once again, innovative ideas have grown and borne delectable fruits for those seated at the C=Hacking table. And once again, in keeping with one of C=Hacking's most hallowed traditions, the issue is really, really late.

A new job, a new move, a new house, a new puppy -- what can I say? This real world stuff takes some getting used to, and so this issue is even later than usual. On the plus side, I think you'll find it worth the wait (at least, I hope so!).

A lot has happened in the C= world since the last issue of C=Hacking. At the personal level, all my moves and uphevals have led to a few changes. Now that I'm fabulously wealthy (compared to my old graduate student salary, I am fabulously wealthy), I splurged on a custom domain name. The Fridge is now located at http://www.ffd2.com, and thus the Official Unofficial C=Hacking Homepage is now located at

http://www.ffd2.com/fridge/chacking/

And in the "well, it's news to me" department:

- SuperCPU acitivty seems to be picking up steam, at long last. The giga.or.at scpu mail list recently revived itself, and several new projects and web pages have appeared.

  Moreover, 65816 assemblers have finally started to appear. El Cheapo Assembler (CheapAss), for SuperCPU-equipped 64s, is available in the Fridge. A new cross-assembler, ACME, is being worked on. And a *third* project, Virtualass816, is said to be underway.

  In addition to the assemblers, there is an 816 backend to LCC, some new operating system projects (e.g. JOS), rumors of games, game fixes (including a Flight Simulator patch by Maurice Randall), and more.

  http://come.to/supercpu is a good place to learn more about new SCPU developments.

- There will be an English version of GO64! magazine, starting with the August issue, available by subscription. For more information, visit http://www.go64.c64.org

- The annual Chicago Expo will take place in -- amazingly enough -- Chicago, on September 25. Last year was a hoot and hopefully our group will be there again this year. For more details, http://members.aol.com/~rgharris/swrap.html will surely contain info as the date draws near.

- Justin Beck, a dj at station KDVS, in Davis, CA, runs a SID radio show every tuesday night at 8PM Pacific time. You can tune in at 90.3 in the Davis/Sacramento area, or the webcast at http://www.kdvs.org. For more information, write jrbeck@ucdavis.edu; actually, write him anyways, to tell him how cool the show is!

- CBM FIDO echos are available at several places on the web these days -- http://cereal.mv.com is a pretty popular source.

- Craig Bruce has been placing old issues of the Transactor online at

  http://www.pobox.com/~csbruce/commodore/transactor/

  Well worth a visit.

- Another unzip program has appeared, by Pasi Ojala. Visit his homepage

  http://www.cs.tut.fi/~albert/

  for more details.

- Richard Atkinson <rga24@hermes.cam.ac.uk> has drawn up schematics of the Commodore Sound Expander, a Yamaha-chip-based cartridge. For more info, email Richard! (And maybe look in a future issue of C=Hacking??? :)

This weekend, here in the states, we are celebrating our independence (I think the Canadians tried to invade us once, or something...). I can't help but reflect that the Commodore 8-bits also represent a kind of independence. As the above (and the below) amply demonstrates, that independent spirit is alive and thriving, a fact that is not only remarkable, but even perhaps worth celebrating.

Enjoy the issue!

-S

....
....
..
.

C=H 18

:::::::::::::::::::::::::::::::: Contents ::::::::::::::::::::::::::::::::

- "VIC-20 Kernel ROM Disassembly Project", by Richard Cini <rcini@msn.com>

  This installment covers interrupts -- IRQ and NMI sources, the kernal handler code.

- "A Diehard Programmer's Introduction to GEOS, and geoWrite Disassembly Notes", by Todd S. Elliott <Eyethian@juno.com>

  As part of his effort to learn about GEOS programming, Todd disassembled geoWrite 128, and patched it to be more accepting of new devices and such. This article summarizes that adventure -- the results and the lessons learned.

- Masters Class: "NTSC/PAL fixing: FLI", by Russell Reed <rreed@egypt.org>, Robin Harbron <macbeth@tbaytel.net>, and S. Judd.

  This time around the subject is FLI and IFLI graphics, and a tutorial on fixing them. Several pictures are provided for the reader to fix, in the included .zip file.

- "Obj3d: The 3D object library", by S. Judd <sjudd@ffd2.com>

  Obj3d is a set of routines for creating, manipulating, and displaying 3D worlds on the 64. It consists of routines to add and remove objects, move and rotate objects, render the display, and so on, and hence vastly simplifies the creation of 3D programs.

  This article actually consists of three articles. The first article describes the library, and walks through a simple example program.

  The second article is the "programmer's guide", containing memory maps and a list of all the routines.

  The third article discusses "stroids", a more advanced example program in which a space ship can fly around a randomly drifting asteroid field, to demonstrate that sophisticated programs can be written with only a few hundred lines of code. The program is purposely left incomplete -- it's up to you to finish it up!

  We had hoped to include a "3D Object Editor", written by Mark Seelye, but it wasn't quite done yet -- next issue!

  Source code and binaries are included at the end of the issue.

..... Credits .....

Editor, The Big Kahuna, The Car'a'carn..... Stephen L. Judd
C=Hacking logo by..... Mark Lawrence

For information on the mailing list, ftp and web sites, send some email to chacking-info@jbrain.com.

Legal disclaimer:

1) If you screw it up it's your own fault!
2) If you use someone's stuff without permission you're a serious dork!

About the authors:

Todd Elliot is 30 years old, working as a Community Client Advisor for the Center On Deafness - Inland Empire, working with deaf people in the local communities. He got his first 64 in December 1983, using it for games, BBS activities, and dabbling in programming. Nowadays Todd focuses on ML programming, and his latest project is an AVI movie player for the SuperCPU. Todd is a huge fan of sports and the Miami Dolphins in particular, and enjoys writing articles and watching movies. At the 1998 Chicago Expo Todd enjoyed meeting all those people who were previously a name and an email address; according to Todd, "The Chicago Expo truly embodies what is going on with today's CBM community."

Richard Cini is a 31 year old vice president of Congress Financial Corporation, and first became involved with Commodore 8-bits in 1981, when his parents bought him a VIC-20 as a birthday present. Mostly he used it for general BASIC programming, with some ML later on, for projects such as controlling the lawn sprinkler system, and for a text-to-speech synthesyzer. All his CBM stuff is packed up right now, along with his other "classic" computers, including a PDP11/34 and a KIM-1. In addition to collecting old computers Richard enjoys gardening, golf, and recently has gotten interested in robotics. As to the C= community, he feels that it

is unique in being fiercely loyal without being evangelical, unlike some other communities, while being extremely creative in making the best use out of the 64.

Pasi 'Albert' Ojala is a 29 year old software engineer, currently working at a VLSI design company on a RISC DSP core C compiler. Around 1984 a friend introduced him to the VIC-20, and a couple of years later he bought a 64+1541 to replace a broken Spectrum48K. He began writing his own BBS, using ML routines for speed, and later wrote a series of demos under the Pu-239 label. In addition to pucrunch and his many C=Hacking articles, Pasi's most recent project is an "unzip" program for the 64. Pasi is also a huge Babylon-5 fan, and has a B5 quote page at http://www.cs.tut.fi/~albert/Quotes/B5-quotes.html

Robin Harbron is a 26 year old internet tech support at a local independent phone company. He first got involved with C= 8-bits in 1980, playing with school PETs, and in 1983 his parents convinced him to spend the extra money on a C64 instead of getting a VIC-20. Like most of us he played a lot of games, typed in games out of magazines, and tried to write his own games. Now he writes demos, dabbles with Internet stuff, writes C= magazine articles, and, yes, plays games. He is currently working on a few demos and a few games, as well as the "in-progress-but-sometimes-stalled-for-a-real-long-time-until-inspiration-hits-again Internet stuff". He is also working on raising a family, and enjoys music (particularly playing bass and guitar), church, ice hockey and cricket, and classic video games.

..... Jiffies .....

Blah.

..... The C=Hallenge .....

Bleah.

..... Side Hacking .....

Data Structures 101 -- Linked lists and a nifty sorting algorithm

by S. Judd and Pasi Ojala --

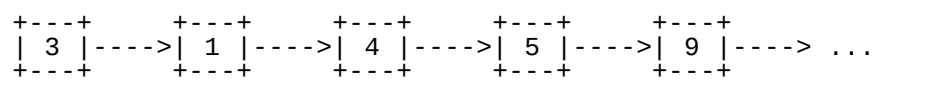
Every computer science major has taken a class in data structures. The 64 programming world, however, is full of non-CS majors. Data structures are such useful tools for programmers to have in their toolbox that I thought it would be a worthwhile thing to have various people periodically review different types of data structures, and common algorithms which make use of them, and their relevance to an 8-bit CBM.

What is a data structure? Perhaps a reasonable definition is that it is an abstract method of storing and retrieving data. These abstractions may then be implemented on the computer, to solve different types of problems. Whereas no single data structure is ideal for all problems, often a given problem has a data structure ideally suited for it. The "optimal" data structure for a given problem depends heavily upon what kind of data is stored, and how it is stored, retrieved, or otherwise manipulated.

(On a more humorous note, /dev/null is ideal for storing information provided you don't need to retrieve it, ever -- this probably doesn't count as a data structure, though).

A simple example of a data structure is an array: abstractly, it stores and retrieves data by means of an index value, e.g. A$(I). On the computer it might be implemented in many different ways, depending on the type of data stored (integers, strings, floating point, custom records, etc.), the dimension of the array, the computer hardware, and even whether the index is e.g. an integer or a string. So it is worthwhile to distinguish the abstract concept of an "array" from its actual implementation.

Another type of useful data structure is a linked list. Imagine attaching a pointer to some chunk of data (usually called a "node"). This pointer can then point, or link, to another chunk of data -- the next node in the list. This can be schematically illustrated as:



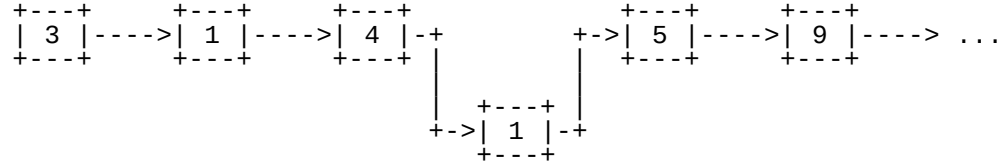
Here each node contains some data (a number) and a pointer to the next node. Starting from the first node -- called the "head" of the list -- a program can reach every other node by following the pointers. This is the basic

idea of a linked list.

Linked lists are used all over the place in the 64. BASIC programs are a type of linked list. Each line of a basic program is stored as a line link, then a line number and the data. The link points to the next line in the program. Having links speeds up BASIC programs (imagine finding and counting the ends of each line for every GOTO or GOSUB), and links are useful because each program line can be of different size (imagine storing a program in something like an array). The end of the list is specified with a line link of 00.

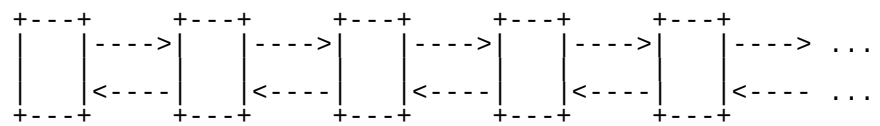
A better example is the DOS. Every 256-byte sector on a disk drive starts with a 2-byte track and sector pointer, pointing to the next sector in the chain, and contains 254 bytes of data. This is exactly a linked list. The directory tells where the first sector of a program is -- the head of the list -- and the program is loaded by simply traversing the list -- following each link and reading the data. The end of the list is specified by using a special link value (track = $FF).

The reason it is "better" is that whereas a BASIC program is stored sequentially in memory, a program stored on disk might be strewn all over the place. This is an important feature of linked lists: they can join together data that is spread all over memory. If you think about it for a moment, you'll quickly realize that elements may be added into (or removed from) the middle of a list simply by changing a pointer:



(Well, okay, two pointers :). Inserting sectors into the middle of a program isn't exactly very useful for disk storage, but there are plenty of applications where it is extremely useful; one such application, discussed below, is sorting data.

Both BASIC and the disk drive are examples of a forward- or singly-linked list. But imagine attaching a second pointer to each disk sector, pointing to the _previous_ track and sector:



This would be a "doubly-linked list". The downside would be a loss of two bytes per sector; the upside would be that if your directory track were destroyed, you could recover all programs on a disk -- in a singly-linked list like C= DOS, you have to know where the start of the list, i.e. the first track and sector, is. Moreover, with a doubly-linked list you can _delete_ a node without even knowing where the node is in the list; with a singly-linked list, you have to search through the list to find the pointer to the node.

Of course, you could add even more pointers to a list, which is done for certain types of trees, for example. (Trees will perhaps be covered some other time).

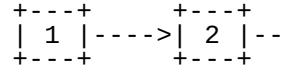
A fast sorting algorithm

Let's say you had a list of numbers that you wanted to sort from largest to smallest. For example, the obj3d library, discussed later in this issue, needs to depth-sort objects, so that far-away objects don't overlap nearby objects when drawn on the screen. This amounts to sorting a list of 16-bit numbers. These numbers are stored in a simple list -- not a linked list, but an array, like:

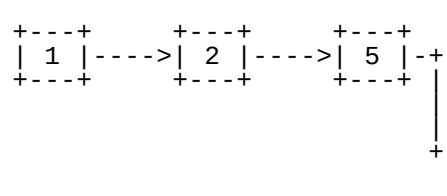
```
lobytes
        lo0
        lo1
        lo2
        ...
hibytes
        hi0
        hi1
        hi2
```

A typical sorting algorithm would have to spend a lot of time swapping numbers, moving stuff around, etc. Even with 16-bits this is a fair amount of overhead, and with even larger numbers it gets very time-

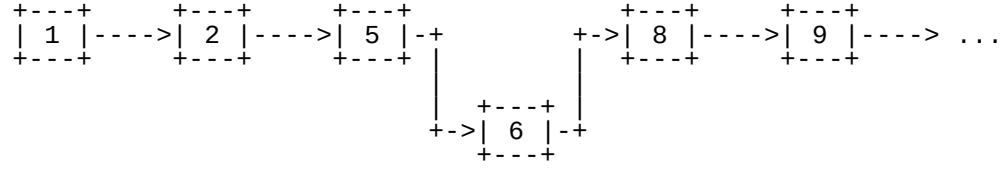
consuming.
        But, as mentioned earlier, a linked list is tailor-made for rearranging data. That is, if we start with a list of sorted numbers, then inserting a new number into the list amounts to finding the right spot in the list,



changing the first part of the list to point to the new number,



and having that new number point to the rest of the list.



So sorting a list of numbers amounts to inserting these numbers into the right spot in the list, one at a time. Amazingly enough, this type of sort is called an "insertion sort".

        Your first thought might be "doesn't that mean a whole lot of overhead in copying data and changing pointers?!" It might, if we were lame PC programmers; but, of course, we are 64 coders, which means we are handsome, witty, superior, humble -- and most of all, sneaky. All we _really_ need is

        a) an index into the list of numbers
        b) a pointer to the next number

The trick is simply to combine the two, by storing the linked list just like the list of numbers, as a sequential array of *8-bit* links:

```
list
        link0
        link1
        link2
        ...
```

Now each link not only points to the next list element -- it exactly points to the next number as well. For example, let's say that the earlier list of numbers was 23,16,513

```
lobyte
        23
        01
        16

hibyte
        00
        02
        00
```

Sorting from largest to smallest should give 1-0-2 -- that is, element 1 is the largest, followed by element 0, followed by element 2. So the linked list could have the form

```
head = 1
list
        2
        0
        $80
```

To see how this works, start with the head of the list, which points to list element 1.

```
        LDX head
```

To get the next element in the list is easy:

```
        LDA list,X
```

Now .X is the current element, and .A is a link to the next element. To traverse the entire list, then we simply need to TAX and loop:

```
        LDX head
:loop   LDA list,X
        TAX
        BPL :loop
```

This will traverse the list, until the $80 is reached. The thing to realize is that .X is *also* a list into the list of numbers, so for example you could print out the numbers, in order, with some code like

```
        LDX head
:loop   LDA lobyte,X
        LDY hibyte,X
        JSR PrintAY
        LDA list,X
        TAX
        BPL :loop
```

Now all we have to do is construct the list!

```
        get next coordinate
        traverse linked list
            compare to coordinates in linked list, to find correct spot
        split linked list in half
        make bottom half of list point to the new element
        make the new element point to the rest of the list
```

So far it is just your basic insertion-sort; if you are familiar with AmigaOS, it is similar to Enqueue(), which inserts nodes by priority. The beauty here is that the index registers make all this really, really easy; here's the sort code from obj3d, to sort positive numbers, stored in CZ and HCZ = lo and hi bytes, from largest to smallest. It starts at the head of the list (at list+$80), traverses the list to find the right spot, and inserts the "node":

```
 :loop  [copy number to be inserted to TEMP, TEMP+1 for a little speed]

         LDY #$80          ;Head of list
:l1      LDA VISOBJS,Y     ;Linked list of objects
         BMI :link
         STY TEMPY
         TAY               ;Next object
         LDA CZ,Y          ;If farther, then
         CMP TEMP          ;move down list
         LDA HCZ,Y
         SBC TEMP+1
         BCS :l1
                           ;Nearest objects last in list
         TYA
         LDY TEMPY         ;Insert into list
:link    STA VISOBJS,X     ;X -> rest of list
         TXA
         STA VISOBJS,Y     ;beginning of list -> X
         DEX
         BPL :loop
:rts     RTS
```

Here, VISOBJS is the linked list of sorted coordinates.

Personally, I think that's awfully nifty -- an insertion sort with almost zero overhead.

In summary, a linked list is simply an abstract method of storing and retrieving data. For certain kinds of problems it is extremely useful, even optimal, and for certain problems it can be implemented in a very efficient and convenient way. And as pointed out in the beginning of this article, that is the essence of a data structure.

I suppose that about wraps things up. So, who wants to volunteer the next data structures article?

    ::

Counting sort
------------- by Pasi Ojala <albert@cs.tut.fi>

Because we got ourselves tangled up with sorting algorithms, we may as well take a look into another one called counting sort. The idea is to sort a list of symbols according not to the symbols themselves, but rather according to a "key" associated with the symbols. This will become clear shortly. This sorting algorithm can be used with integer-valued data when the range of sorting key values (the values by which the order is decided) is small enough.

The foundation for counting sort is to make one pass through the data and determine the sorted order of the data, then make another pass and perform the actual shuffling of data.

Now, what kind of a data structure is used for counting sort? Surprise, we only need a counter for every possible sorting key. This is why a limited range of values is required or the memory consumption would simply be too large. And we need an array of input data and an array for the sorted output data, as the sorting can't be performed in place.

An example sorts the following data, which in fact represents a Huffman tree. The sorting keys are the code lengths, and the associated data is the corresponding symbol. The sorting of symbols is required in some of the Huffman tree creation routines.

```
key: 4 3 1 5 3 6 3 6
sym: A B C D E F G H
```

The sorting keys can have value from 1 to 6, so we need 6 counters. Their initial values are set to zero:

```
for(i=1;i<7;i++)
    count[i-1] = 0;

count: 0 0 0 0 0 0
```

Then we perform the counting stage by going through the data and incrementing the counter corresponding to the sorting key value. In the end the counters contain the number of each code length found in the data.

```
for(i=0;i<#ofvalues;i++) {
    count[sort_key[i]-1] = count[sort_key[i]-1] + 1;
}

count: 1 0 3 1 1 2
```

Then cumulative counts are calculated for this array.

```
for(i=1;i<6;i++) {
    count[i+1] = count[i+1] + count[i];
}

count: 1 1 4 5 6 8
```

If you take a close look into the meaning of the values now in the count array, you might notice that the last count value gives us from which element downward to stick the data with sorting key 6, the previous one where to stuff data with key 5 and so on.

So, next we simply copy each element in the data into its final place in the output array using the cumulative counts calculated previously. Note that in C the indexing starts from 0, but in the table from 1 for the reader's convenience.

```
for(i=#ofvalues-1;i>=0;i--) {
    count[key[i]-1] = count[key[i]-1] - 1;

    outputkey[count[key[i]-1]] = key[i];
    outputsym[count[key[i]-1]] = sym[i];
}
```

```
    1 2 3 4 5 6   1 2 3 4 5 6 7 8
---------------------------------
6   1 1 4 5 6 8   x x x x x x x 6
H             7   X X X X X x X H

3   1 1 4 5 6 7   x x x 3 x x x 6
G       3         X X X G X x X H

6   1 1 3 5 6 7   x x x 3 x x 6 6
F             6   X X X G X x F H
```

```
        3   1 1 3 5 6 6   x x 3 3 x x 6 6
        E       2         X X E G X x F H

        5   1 1 2 5 6 6   x x 3 3 x 5 6 6
        D           5     X X E G X D F H

        1   1 1 2 5 6 6   1 x 3 3 x 5 6 6
        C   0             C X E G X D F H

        3   0 1 2 5 6 6   1 3 3 3 x 5 6 6
        B       1         C B E G X D F H

        4   0 1 1 5 6 6   1 3 3 3 4 5 6 6
        A         4       C B E G A D F H
```

So, after the loop we have sorted the symbols according to the code lengths. We haven't talked about the O() notation, but the counting sort is an O(n) process. Counting sort only needs three fixed-sized loops, thus if the amount of data increases linearly, the sorting time also increases linearly. With insertion sort the sorting time increases geometrically because searching for the right spot to add a node takes longer and longer when the number of already sorted nodes increases.

One extra bonus for counting sort is that it preserves the so-called inside order. The order of elements with identical keys is preserved in the end result. This is very important in many algorithms, like the Huffman tree creation GZip uses (and which gunzip.c64 used before the algorithm was changed to not require sorting).

On the minus side is the necessity of the output buffer. Other sorting algorithms can sort the data in-place, although that also means that the data must be copied a lot of times.

The preservation of inside-order also allows the sorting of data by multiple keys: first sort by secondary keys, then by primary keys. After this the data is sorted by the primary keys and all elements having identical keys are sorted by the secondary keys.

And that's it!

....
....
..
.

::

Main Articles

VIC KERNAL Disassembly Project - Part II
Richard Cini
May 21, 1999

Introduction

Last issue, we began by examining the VIC's start-up sequence. We began with the processor jump at power-on to location $FFFC upon power-up and ended at the Kernal’s jump into the BASIC ROM.

This issue, we will look at the two other hard-coded processor vectors, the IRQ and NMI vectors. Just like the RES* pin on the 6502 processor, the IRQ* and NMI* pins (the hardware portion of the interrupt) are hard-coded to two memory locations, $FFFE and $FFFA, respectively. When the processor senses a negative-going pulse on these pins of at least 20uS in duration, the processor will jump to the respective vector locations (the software portion of the interrupt, also called an "interrupt handler"):

```
FFFA            ;================================
FFFA            ; - Power-on Vectors
FFFA            ;
FFFA A9 FE      .dw NMI                         ;$FEA9
FFFC 22 FD      .dw RESET                       ;$FD22
FFFE 72 FF      .dw IRQ                         ;$FF72
```

Both the IRQ* and NMI* lines are available at the expansion connector at the back of the VIC. They are also connected to the IRQ* output lines on the VIC's two 6522 VIA chips (just like the 64); the IRQ* is connected to VIA2 and the NMI* to VIA1.

Hardware interrupts on the VIC-20 may be triggered by any one of several events which is recognized by either one of the VIC's two VIA chips, or by an external event which triggers the IRQ* or NMI* pins on the VIC's expansion connector. The software IRQ routine (at $FF72) is also entered by the execution of the BRK processor instruction.

The VIA chips are capable of generating a hardware interrupt based on the events as outlined in register $911D, as described in the section titled "VIA Registers Used in Following Code." In general, interrupt events result from the VIA's internal timers reaching 0 and active transitions in the logic level of the CA1/CB1/CA2/CB2 handshaking pins.

Some of the timers and handshaking pins are programmed or connected to hardware within the VIC. For example, timer 1 on VIA2 is initialized to provide a periodic interrupt every 0.015 seconds, or an approximate rate of 65 Hz. This interrupt is used to update the time-of-day clock and to provide timing pulses for cassette tape operations. This timer sharing is the main reason behind the jiffy clock losing time during tape operations. Similarly, the processor NMI* line is triggered by the IRQ* output on VIA1.

In summary, by default, the interrupt handler code is executed as a result of the following:

For the IRQ code: the 60Hz periodic interrupt from Timer 1 on VIA2 and the processor BRK instruction.

For the NMI code: pressing of the RESTORE or Run/Stop-RESTORE key combinations and RS232 operations.

The IRQ and NMI software routines could also be triggered if the programmer makes use of some of the other VIA pins for a home-brewed project. For example, if a home-brew alarm system is configured to trigger the CA2 pin when there is motion in the back yard. The user's program could be set up to watch for an interrupt generated by the CA2 pin, to sound a siren.

VIA Registers Used in the Following Code

The 6522 VIA has 16 byte-wide registers which control the functioning of the chip or enable the receipt or transmission of data on the I/O pins. For the sake of space, we'll only discuss the registers directly applicable to this issue's code.

$9111 is the Port A output register, an 8-bit I/O port which automatically uses the CA1 and CA2 pins for handshaking. Port A is duplicated at $911F, but changes to the register do not affect CA1 and CA2 (so, no handshaking). In the above code, only BIT7 and BIT6 are relevant.

The bitfields are:

BIT7	IEEE ATN out
BIT6	Cassette sense switch
BIT5	Joystick fire button
BIT4	Joystick left
BIT3	Joystick down
BIT2	Joystick up
BIT1	IEEE Serial Data In
BIT0	IEEE Serial Clock In

$911D is the register (the Interrupt Flag Register, "IFR") that indicates which event triggered the interrupt. BIT3, BIT2, and BIT0 are left unprogrammed in the above code, but are shown below for completeness. BIT4 manages the RS232 receive function, and BIT1 is connected to the RESTORE key.

	SET BY	CLEARED BY
BIT7	NMI status (set when any of the lower bits are set)	
BIT6	Timer 1 time-out	read T1 LW and wrt T1 HW latch
BIT5	Timer 2 time-out	read T2 LW and wrt T1 HW latch
BIT4	CB1 transition	R/W Port B
BIT3	CB2 transition	R/W Port B
BIT2	Completion of 8 shifts	R/W shift register
BIT1	CA1 transition	R/W Port A ($9111 only)
BIT0	CA2 transition	R/W Port A ($9111 only)

According to the 6522 data sheets, the shift register interrupt is an allowed interrupt, but there is a bug in early 6522s that prevented the shift register from working properly. In the VIC, the shift registers are disabled by default and remain unused in the KERNAL. It is possible for a user program to enable the shift registers and use them, but the quality of the results would be lacking because of the bug.

$911E (the Interrupt Enable Register, "IER") is the register that enables or prevents the lower six bits in the IFR from triggering an interrupt. Writing a 0 to BIT7 clears the bits in the IER according to the bit pattern in the lower six bits. Writing a 1 to BIT7 sets the IER according to the bit pattern. For example, if one wanted to enable all of the above as sources of interrupts, one would write %11111111 to the IER. Disabling all of the above as interrupt sources would be accomplished by a write of %01111111 to the IER.

$9110 is the I/O register for Port B of VIA1. This 8-bit port is connected to pins PB0-PB7 of the VIC User Port. When the RS232 interface module is connected to the User Port, Port B doubles as the RS232 port, with the following bitmap:

```
PB7      Data Set Ready (DSR) in
PB6      Clear To Send (CTS) in
PB5      [no connection]
PB4      Data Carrier Detect (DCD) in
PB3      Ring Indicator (RI) in
PB2      Data Terminal Ready (DTR) out
PB1      Request To Send (RTS) out
PB0      Receive Data
```

CB1 acts as the interrupt source for the NMI receive routine and is physically connected (externally) to PB0 so that a received bit of data triggers an interrupt. CB2 acts as the RS232 transmit line.

Register $9114 is an 8-bit register which holds the least-significant byte of Timer 1's countdown value.

Registers $9118 and $9119 are the least-significant and most-significant bytes of Timer 2's countdown value. Together, they provide a 16-bit countdown capability for use as the baud rate clock.

Register $911C is called the Peripheral Control Register, the "PCR". The PCR controls how the four handshaking lines (CA1/2 and CB1/2) act. On VIA1, CB2 is connected to the RS232 transmit line, and CA2 is connected to the cassette tape motor control circuitry. Both CA2 and CB2 have eight possible modes that can be manual or automatic, positively or negatively triggered, input or output oriented. Input modes set flags in the IFR based on programmed transitions. When programmed as outputs, CA2/CB2 lines are triggered based on writing to Port B.

NMI Processing

Let's look at the NMI routine first. The NMI is triggered through the use of the RESTORE key connected to the CA1 line, the CB1 RS232 receive data line, and the expiration of Timer 1 and Timer 2, which manages framing for RS232 transmit and receive operations, respectively.

The NMI code begins with some stub code that enables the redirection of the NMI processing if so programmed by a user. The indirect jump at $FFEA is similar to chainable interrupt processing on MS-DOS based computer systems -- save the old pointer, redirect pointer to your own code, and then chain to the system NMI processor. Several other Kernal vectors are handled in the same way: IRQ, Break, Open, Close, Load, Save, Set Input, Set Output, Input, Output, Get Character, and Clear I/O Channel.

```
FEA9            ;================================================
FEA9            ; NMI - NMI transfer entry
FEA9            ;================================================
FEA9             NMI
FEA9 78                  SEI             ; disable interrupts
FEAA 6C 18 03            JMP (NMIVP)     ;$FEAD allows redirection of NMI
```

The actual NMI processing code follows the redirect stub. The NMI routine is broken into three parts: RESTORE processing, RS232 transmit management and RS232 receive management.

```
FEAD            ;================================================
FEAD            ; LNKNMI - Link to actual NMI code. The first
FEAD            ;       part manages the R/S-R keys
FEAD            LNKNMI
FEAD 48                  PHA             ; save registers .A
FEAE 8A                  TXA
FEAF 48                  PHA             ; .X
```

```
FEB0 98                 TYA
FEB1 48                 PHA             ; .Y
FEB2 AD 1D 91           LDA D1IFR       ;check VIA1 interrupt flag register
FEB5 10 48              BPL WARMEOI     ;no flag present, then issue EOI

FEB7 2D 1E 91           AND D1IER       ;see if found interrupt is allowed
                                        ; based on IER mask
FEBA AA                 TAX             ;save bitmask of enabled and active
                                        ; interrupts
FEBB 29 02              AND #%00000010  ;VIA1/CA1 RESTORE key?
FEBD F0 1F              BEQ WARM1       ;not RESTORE, so move on to RS232

                                        ;Got here, so NMI must be RESTORE
                                        ; key
FEBF 20 3F FD           JSR SCNROM      ;scan for A000 ROM (covered last
                                        ; issue)
FEC2 D0 03              BNE LNKNMI1     ;no ROM at $A0, so skip ROM NMI
                                        ; routine

FEC4 6C 02 A0           JMP (A0BASE+2)  ;jump to ROM NMI routine

FEC7            LNKNMI1                 ;continue NMI processing.
FEC7 2C 11 91           BIT D1ORA       ;test ATN IN(7)/cass switch (6)
                                        ; bits
FECA 20 34 F7           JSR IUDTIM      ;update TOD clock
FECD 20 E1 FF           JSR STOP        ;check for STOP key Z=1 if STOP
                                        ; pressed
FED0 D0 2D              BNE WARMEOI     ;no stop key, so skip vector
                                        ; restore, VIC and I/O
                                        ; initialization. Go EOI.
```

We reach the following code block if Run/Stop is pressed along with the RESTORE key. The code results in a soft reset: it restores the default system vectors, initializes the I/O chips to their default, and initializes the screen editor. The routine then jumps into BASIC, bypassing BASIC's normal initialization routines, leaving program RAM undisturbed.

```
FED2            ;===============================================
FED2            ; WARMST - Default USER vector
FED2            ;          Also get here from BRK instruction (through IRQ
FED2            ;          vector) or RUN/STOP-RESTORE key combination
FED2            WARMST
FED2 20 52 FD           JSR IRESTR      ;restore default vectors (covered
                                        ; last issue)
FED5 20 F9 FD           JSR IOINIT      ;initialize I/O (covered last
                                        ; issue)
FED8 20 18 E5           JSR CINT1       ;initialize screen editor (covered
                                        ; last issue)
FEDB 6C 02 C0           JMP (BENTER+2)  ;jump to BASIC NMI routine
```

If program execution gets here without passing through the WARMST code, then the NMI resulted from an RS232 event, such as when the transmit or receive timers time-out (signaling that the VIC is done transmitting or receiving a character of information). For now, we'll skip the actual RS232 transmit/receive code; it'll be covered in a later issue.

Since the Kernal programmers could not reliably use the shift registers in the VIAs, it appears that they synthesized a shift register in the NMI routine. This emulation enables the RS232 code to work properly.

The code continues with the NMI part of the RS232 transmit and receive character processing routine:

```
FEDE            WARM1
FEDE AD 1E 91           LDA D1IER       ;get IER bitmap
FEE1 09 80              ORA #%10000000  ;set mask for enabling interrupts
                                        ; according to existing bitmap
FEE3 48                 PHA             ;save "enable" bitmap
FEE4 A9 7F              LDA #%01111111  ;mask-disable all interrupts on
                                        ; VIA1
FEE6 8D 1E 91           STA D1IER       ;go do it

FEE9 8A                 TXA             ;restore mask for active interrupts
FEEA 29 40              AND #%01000000  ;IFR bit6 =TIMER1 time-out (RS232
                                        ; clk)
FEEC F0 14              BEQ WARM2       ;T1 done, go to RS232/RX chr

FEEE A9 CE              LDA #%11001110  ;set/reset bit5 to xmit char
FEF0 05 B5              ORA NXTBIT      ;RS232 transmit - next bit to send;
FEF2 8D 1C 91           STA D1PCR       ;CB2 manual L/H; CB1 neg trans for
                                        ; IRQ; CA2 manual H; CA1 neg trans
```

```
                                        ; for IRQ; CB2=TX, CA2=cass motor
                                        ; control
FEF5 AD 14 91           LDA D1TM1L      ;get VIA1/T1 count low byte
FEF8 68                 PLA             ;restore IER bitmap...
FEF9 8D 1E 91           STA D1IER       ; ...and save it
FEFC 20 A3 EF           JSR SSEND       ;send RS232 char
FEFF
FEFF            WARMEOI
FEFF 4C 56 FF           JMP EOI         ;end of interrupt

FF02            WARM2                   ;RS232 receive NMI routine
FF02 8A                 TXA             ;restore IFR mask from above
FF03 29 20              AND #%00100000  ;VIA1/T2 time-out (done receiving
                                        ; character from RS232 channel)?
FF05 F0 25              BEQ WARM3       ;yes, so move byte to buffer

                                        ;collect bits...
FF07 AD 10 91           LDA D1ORB       ;get user port bitmap
FF0A 29 01              AND #%00000001  ;bit0=RS232/RX
FF0C 85 A7              STA INBIT       ;save received bit
FF0E AD 18 91           LDA D1TM2L      ;get VIA1/T2L count
FF11 E9 16              SBC #$16        ; subtract 22d
FF13 6D 99 02           ADC BAUDOF      ; add low word of bit transmit time
FF16 8D 18 91           STA D1TM2L      ; save it
FF19 AD 19 91           LDA D1TM2L+1    ;get VIA1/T2H count
FF1C 6D 9A 02           ADC BAUDOF+1    ; add high word of bit xmit time
FF1F 8D 19 91           STA D1TM2L+1    ; save it
FF22 68                 PLA             ;restore old IFR bitmap...
FF23 8D 1E 91           STA D1IER       ; ...and save it
FF26 20 36 F0           JSR SERRX       ;signal RS232 receive routine
FF29 4C 56 FF           JMP EOI         ;end of interrupt

FF2C            WARM3                   ;received new char, so buffer it
FF2C 8A                 TXA
FF2D 29 10              AND #%00010000  ;CB1 interrupt (RX data bit
                                        ; transition)
FF2F F0 25              BEQ EOI         ;no bit, exit
```

One interesting fact about the VIC is that originally it was supposed to include a 6551 ACIA (RS-232) communications chip as standard equipment.

However, when MOS could not supply an adequate number of working chips to support the anticipated VIC production volume, the VIC engineers decided to emulate the 6551 in software. The VIC Programmer's Reference Guide makes mention of the 6551 registers, but the VIC clearly does not contain a 6551. See Cameron Kaiser's Commodore Knowledge Base for more details. The URL is http://calvin.ptloma.edu/~spectre/ckb/

The 6551 pseudo-Control Register contains the stop bits, word length, and baud rate parameters. The pseudo-Command Register contains the parity, duplex, and handshaking parameters. The pseudo-Status Register contains a result code bitmap. After setting the baud rate divisor, the routine updates the number of bits to transmit and exits.

```
FF31 AD 93 02           LDA M51CTR      ;pseudo 6551 control register
FF34 29 0F              AND #%00001111  ;pass the baud rate parameter only
FF36 D0 00              BNE $+2         ;I/O delay
FF38 0A                 ASL A           ;shift left
FF39 AA                 TAX             ;save shifted baud rate bitmask and
                                        ; use as an index into a data table
                                        ; with the receive timer values
FF3A BD 5A FF           LDA R232TB-2,X  ;index into baud rate divisor table
FF3D 8D 18 91           STA D1TM2L      ; and save the divisor into the VIA

FF40 BD 5B FF           LDA R232TB-1,X  ; timer count register
FF43 8D 19 91           STA D1TM2L+1

FF46 AD 10 91           LDA D1ORB       ;read RS232 output register
FF49 68                 PLA             ;restore IFR bitmap
FF4A 09 20              ORA #%00100000  ;T2 interrupt flag
FF4C 29 EF              AND #%11101111  ;pass T2 int. flag but not CB1
FF4E 8D 1E 91           STA D1IER       ;save new interrupt bitmap
FF51 AE 98 02           LDX BITNUM      ;get total number of bits to TX/RX
FF54 86 A8              STX BITCI       ;save as receiver bit count
FF56            ;
FF56            ; EOI - End of Interrupt
FF56            ;
FF56            EOI
FF56 68                 PLA             ;restore registers
FF57 A8                 TAY
FF58 68                 PLA
FF59 AA                 TAX
```

```
FF5A 68                 PLA
FF5B 40                 RTI             ;return from interrupt
```

IRQ Processing

The IRQ routine is another very important routine for the VIC, as various housekeeping chores are performed during the interrupt. The processor BRK instruction also points to the IRQ vector, so there is some testing early in the routine to handle the BRK instruction.

The entry code is similar to the NMI entry code, and similarly, allows function chaining. For example, one could write a small IRQ routine to provide keyboard "click" feedback, with the code activated during IRQ processing.

```
FF72                ;================================================
FF72                ; IRQ - IRQ transfer point
FF72                ;================================================
FF72                IRQ
FF72 48                     PHA             ; save .A
FF73 8A                     TXA
FF74 48                     PHA             ; save .X
FF75 98                     TYA
FF76 48                     PHA             ; save .Y
FF77 BA                     TSX             ; get stack pointer
FF78 BD 04 01               LDA FBUFFR+4,X  ;look in stack for PSW BRK flag
FF7B 29 10                  AND #%00010000  ;bit4 of PSW; breakpoint or IRQ?
FF7D F0 03                  BEQ BRKSKIP     ;IRQ, branch
FF7F
FF7F 6C 16 03               JMP (BRKVP)     ;jump to breakpoint processor,
                                            ; which is the WARMST location.
FF82                BRKSKIP
FF82 6C 14 03               JMP (IRQVP)     ;jump to normal IRQ routine at
                                            ; $EABF
```

When the code is finished determining if the interrupt was triggered by a timer tick interrupt or through a BRK instruction, the IRQ code continues at $EABF. If it is a BRK instruction, code execution continues at the WARMST location. The rest of the IRQ code calls the clock update routine, handles blinking the cursor, tape motor control, and scanning the keyboard -- all functions that could be considered "user interface" functions.

```
EABF                ;================================================
EABF                ; IRQVEC - IRQ Vector
EABF                ;
EABF                IRQVEC
EABF 20 EA FF               JSR UDTIM       ;update system clock FFEA=>F734
EAC2 A5 CC                  LDA BLNSW       ;cursor enable (0=enable)
EAC4 D0 29                  BNE IRQVEC2     ;non-zero, so skip blink code
EAC6
EAC6 C6 CD                  DEC BLNCT       ;decrement blink count
EAC8 D0 25                  BNE IRQVEC2     ;not reached 0, so move on to
                                            ; cassette timing stuff
EACA A9 14                  LDA #$14        ;reset blink timer to 20d (ms)
EACC 85 CD                  STA BLNCT       ;save new count
EACE A4 D3                  LDY CSRIDX      ;get cursor column
EAD0 46 CF                  LSR BLNON       ;get blink phase into C flag
EAD2 AE 87 02               LDX CSRCLR      ;get color under cursor
EAD5 B1 D1                  LDA (LINPTR),Y  ;get character code of current char
EAD7 B0 11                  BCS IRQVEC1     ;blink already on, so continue
EAD9
EAD9 E6 CF                  INC BLNON       ;increment blink on flag
EADB 85 CE                  STA GDBLN       ;save char under cursor
EADD 20 B2 EA               JSR CCOLRAM     ;get pointer to color RAM
EAE0 B1 F3                  LDA (COLRPT),Y  ;get color code for cursor location
EAE2 8D 87 02               STA CSRCLR      ;save it
EAE5 AE 86 02               LDX CLCODE      ;get color under cursor
EAE8 A5 CE                  LDA GDBLN       ;get char again
EAEA
EAEA                IRQVEC1
EAEA 49 80                  EOR #%10000000  ;update cursor with blink phase
EAEC 20 AA EA               JSR PRNSCR1     ;set char and color
EAEF
EAEF                IRQVEC2                 ;scan for tape switch pressed
EAEF AD 1F 91               LDA D1ORAH      ;read I/O bitmap
EAF2 29 40                  AND #%01000000  ;cassette switch pressed?
EAF4 F0 0B                  BEQ IRQVEC3     ;no, turn motor off
EAF6
```

```
EAF6 A0 00              LDY #$00        ;set motor "on"
EAF8 84 C0              STY CAS1        ;clear motor interlock flag
EAFA AD 1C 91           LDA D1PCR       ;get PCR bitmap
EAFD 09 02              ORA #%00000010  ;set motor control bit to "on"
EAFF D0 09              BNE IRQVEC4     ;go to timer1 test
EB01
EB01              IRQVEC3               ;set motor to "off"
EB01 A5 C0              LDA CAS1        ;get cassette interlock flag
EB03 D0 0D              BNE IRQVEC5     ;is flag 1, then exit-motor is off
EB05
EB05 AD 1C 91           LDA D1PCR       ;get PCR bitmap
EB08 29 FD              AND #%11111101  ;set motor control bits to "off"
EB0A
EB0A              IRQVEC4
EB0A 2C 1E 91           BIT D1IER       ;IER bit7=IERs/c bit6=T1
EB0D 70 03              BVS IRQVEC5     ;is timer 1 still enabled? Yes,
                                        ; skip update
EB0F
EB0F 8D 1C 91           STA D1PCR       ;set motor status
EB12
EB12              IRQVEC5
EB12 20 1E EB           JSR ISCNKY      ;scan keyboard
EB15 2C 24 91           BIT D2TM1L      ;D2T1 latch LSB bits 7-6

EB18 68                 PLA             ;restore registers and return from
EB19 A8                 TAY             ; interrupt
EB1A 68                 PLA
EB1B AA                 TAX
EB1C 68                 PLA
EB1D 40                 RTI
```

Conclusion
==========

The VIC's NMI and IRQ routines are important to the smooth operation of the VIC, handling a few critical functions that make an impact on usability. The NMI handler deals with soft-reset processing and RS232 communications timing, and the IRQ handler deals with the cursor, the keyboard, and cassette deck timing. The division of labor between the two routines is a sensible one. The routines are relatively compact, but allow expansion through chaining.

Next time, we'll examine more routines in the VIC's KERNAL, including some of the routines called from NMI and IRQ.

....
....
..
.

C=H #18

::

geoWrite Disassembly Notes
By Todd S. Elliott - Eyethian@juno.com

Introduction
============

As part of an effort to learn GEOS programming and to improve an application I was working on, I disassembled geoWrite 128, and along the way, I made several modifications to make it work with modern day GEOS systems. This article contains the fruits of my efforts and some of the lessons I learned. The first part of the article describes largely the territory that comes with GEOS programming. The second part discusses my particular disassembly procedure, and the last part contains the actual disassembly notes to geoWrite 128 v2.2.

Background
==========

When GEOS came out in mid-1980's, I tried it and it was cumbersome and slow. I took an immediate dislike to it and sparingly used it, as if it were a demo. But, CMD arrived with their powerful peripherals and I began to enjoy using GEOS. Little did I realize it, GEOS insidously has made me a convert and a believer in the fact that a c64/128 can do a GUI OS.

But, I was still a user all of this time, and I'll admit that I didn't think about programming for it and was a little bit intimidated. But, the WWW craze hit, and there were browsers popping up everywhere. All of a sudden, scarce

GEOS information was made immediately available, including programming docs. I thought to myself, why not try GEOS programming? All I needed was an idea. Naively, I thought I would try to create a graphical (mono) browser that could read in HTML files off a local system, and thought that GEOS 128 was a natural fit. Thus, 'Constellation' was born, nutured and died an untimely death, like so many of my other projects before and after then, and they shall remain nameless. :(

First off, I had to use geoProgrammer and it was not an easy package to use and master, and its manual was the heaviest in the business. Secondly, GEOS uses an event driven nature of programming and that goes against my beliefs in programming for the CBM 8-bit machines for years. Third, there were a lot of system calls at my disposal and I really didn't know how to use them effectively, or used wrong calls, and Berkeley Softwork's own programming docs had inaccuracies and incompleteness. Fourth, GEOS 128, while a marvel and a technological breakthrough, it was too limited, in my view, for a serious application such as a mono graphical browser.

My main obstacle, as one would strangely call it, to GEOS programming is the Berkeley Softwork's (BSW) own programming suite, geoProgrammer. One main feature that set it apart from other assemblers was that it had a linker. This was a new concept for me and took a while to warm up to the idea. (BTW, Merlin is the only other assembler that I know of which uses a linker.) Next, Berkeley Softworks added a host of features, directives, psuedo-ops and even included a very powerful symbolic debugger in this programming suite. This made programming for GEOS a complex task and I really had to keep tabs on my projects and track my GEOS development. This complexity was only exacerbated by several bugs that made GEOS programming an exactingly task.

But if used correctly, geoProgrammer does a very good job in creating GEOS programs easily and quickly. Fortunately, Concept, Maurice Randall's integrated development environment for Wheels, comes to my rescue by fixing most of the problems plaguing geoProgrammer and boosting productivity in programming for GEOS. Despite Concept's ease of use and its integrated development environment, one still has to know geoProgrammer in order to use it effectively. This is because Concept has an integrated development environment in which it acts as a desktop and allows the programmer to select geoWrite, geoAssembler or geoLinker and work on source code files. Secondly, Concept does not work in GEOS and is specific to the Wheels upgrade. For more information, download Concept at:

http://people.delphi.com/arca93/files/concept.wr3

There are other development packages available for GEOS programming, and I won't go into them as I never have used them but will name them for the sake of completeness. There was a program called geoCOPE, which was reviewed in the Transactor and was billed as a simple alternative to geoProgrammer for novices who desire to learn a little bit about GEOS programming. Then there's the MegaAssembler, a german GEOS programming suite which is supposedly powerful and complex as geoProgrammer. I do not know if it is in the public domain or if it has an English version.

Once I got past the intricacies of geoProgrammer, I was stymied at first by the event-driven nature underlying GEOS. I was conditioned to create a main loop checking for user input and do various activities as necessary. The opposite is used in GEOS. I had to GIVE up control of the application to the GEOS main loop and let it do its job. That meant I had to set up the icons, the menus, the screen interface, etc. and do an RTS. The GEOS main loop then takes over and if an icon was clicked upon, for example, my application finally takes control again and acts upon the icon click. As I got used to it, I began to appreciate the event-driven nature of GEOS and how it has made my job programming for GEOS that much easier.

The event-driven nature of GEOS is backed up by a phalanx of GEOS system calls, of which most can be accessed by the programmer without the need to ever use the KERNAL at all. Some system calls can be quite complex and can basically take over the entire computer or can be very destructive if used the wrong way. To make matters worse in trying to understand this entirely new graphical OS, BSW's own programming docs were incomplete and had inaccuracies. Despite that, both programming docs, the Official GEOS Programmer's Reference Guide and the Hitchhiker's Guide to GEOS contains a lot of useful information and when taken together, is truly the 'bible' on GEOS programming. Anyone wishing to do serious GEOS programming needs both books.

Still, I managed to overcome these obstacles to GEOS programming and actually created a user interface for Constellation. But I ran into a serious limitation of GEOS 128; it had no memory management routines. I required that Constellation be able to handle HTML files ranging from 254 bytes to 2Mb monsters with equal aplomb. GEOS 128 has system calls for using REU memory, but I was not very familiar with using them and there was no way of telling

how GEOS 128 used expansion memory. Secondly, I wasn't too sure on how to display text and graphics with ease and whatever methods that I looked at, it simply looked too cumbersome and difficult.

However, there was a 'browser' already in the GEOS environment; users simply do not view it as that way. It's geoWrite, and it reads and mixes in text and graphics with ease. Secondly, it is a full-blown BSW application that works seamlessly with GEOS as opposed to Constellation's often mysterious crashes. At that point, I decided to scuttle Constellation and open a dissection into the inner workings of geoWrite 128 and use that knowledge to revive Constellation. And as a bonus, I would learn a lot more about GEOS programming far better than what BSW's own docs could provide. Only that it exactly hasn't turned out that way so far...

As I progressed into disassembling geoWrite 128 v2.1, I came across several sections of code and said to myself, 'I could change this or that'. There were code that mostly dealt with file handling, and this stuff was obsolete by the time high powered GEOS systems came into fruiton. For example, BSW put in code that merely 'toggled' the data device, and at that time, it was sensible as two drive systems were possible. But today's GEOS systems can support up to four drives and this code is an annoyance and limits geoWrite 128's usability in modern day GEOS systems.

So, I disassembled geoWrite 128 with the full intention of porting such GEOS knowledge to Constellation and wound up improving geoWrite 128 instead. Here's what I improved so far in geoWrite 128 v2.2:

* Complete four drive support in its file requestors.
* Capable of booting up from any drive (A-D) when a datafile is clicked upon. Improved booting up sequence.
* Loads completely into expansion memory in Wheels equipped systems. In this case, geoWrite 128 no longer needs to access the disk device to retrieve its own modules, and simply fetches them from the faster RAM expansion. A very useful feature for users who only have 41's, 71's, 81's or FD's. This feature does not work in GEOS 128 because there is no reliable way of knowing how it manages RAM expansion.
* Semi-intelligent - Does not prompt the user to insert a new disk if non-removable media is used.
* Displays the DISK icon when a Native ramdisk is used. Previously, geoWrite 128 would not display a DISK icon (used to change disks or partitions) when a ramdisk is accessed. But native ramdisks have their own subdirectories and the DISK icon is activated accordingly.
* Also displays the DISK icon if a ramlink 1581 partition is used.
* The DISK icon is also displayed in the disk device from which geoWrite 128 was loaded from, because geoWrite 128 is now 100% ram resident.
* Autodetects whether it is running in GEOS 128 v2.0 or Wheels 128 systems.

The rest of this article mainly focuses on how geoWrite 128 v2.2 is constructed, how to disassemble a GEOS program, and mostly tries to offer inspiration for others to undertake GEOS programming endeavors. There is so much that needs to be done and can be done in GEOS. Without further ado... Onward!

Disassembly For Dummies. :)

Before we begin with the disassembly notes, let's describe how I conducted the disassembly of geoWrite 128 v2.1. In fact, a program called geoSourcer (64/128 versions) was recently released into the public domain while I was busy disassembling geoWrite 128 v2.1. Since I haven't disassembled seven other VLIR modules, I plan to use geoSourcer to do the job and will post a review, tutorial or critique of sorts either here in a future issue of C=Hacking or on comp.sys.cbm.

First, I personally use the DarkStar suite of disk utilities (ZipCode, etc.) and use its excellent disk editor. The reason is that I need to 'link' or 'de-link' the individual VLIR modules from the rest of the program for later disassembly or reassembly. One must have knowledge of how VLIR files are structured at this stage of disassembly.

There are two formats under GEOS; the sequential format (not to be confused with SEQ files) and the VLIR format. The sequential format simply consists of a file running in a contigious fashion and is loaded in at once. The VLIR format is similar to CBM REL files in which there are 127 individual records, and each record can basically be of any length. VLIR files are broken into records, or if in program files, 'modules'. With respect to programs and not datafiles, VLIR #0 will be the main module which is loaded and executed first in GEOS. In turn, this module will load in other modules in its VLIR file structure whenever necessary. This allows for much larger GEOS programs to run and still have room to spare. For example, geoWrite 128 is 35K, and if it were a sequential file, there would be no room left over for the actual

datafile to coexist. What about geoPublish at 99K? The VLIR file format is the only way geoPublish can run in GEOS and on the CBM 8-bit platform. In essence, the VLIR format allows GEOS to use the disk device as virtual memory device by loading in modules as needed.

geoWrite 128 v2.1 consists of eight VLIR files. VLIR #0 loads in at $0400, and contains the main 'meat' of the program, with icons, menus, dialog box info, etc. VLIR #1 contains mostly screen setup routines, the geos kernal ID check, checking the printer driver and other initialization routines. VLIRs #2 through #4 are unknown at this point, while VLIR #5 contains the main routines for datafile handling. VLIR #6 is unknown, while VLIRs #7 and #8 contain the printing code. VLIRs #1 through #7 all load in at $3244, and VLIR #8 loads in at $0680. I focused on VLIR #0 and #5.

To disassemble the VLIRs, I located the record on the disk and created a separate directory entry pointing to the record as a PRG (non-GEOS) file. I looked at the first two bytes of the file. They are actual pieces of code, but outside of GEOS, they are mistaken for as load addresses. I wrote down these two bytes somewhere so I could later retrieve them. At this point, I have to know its true load address. Let's say that it's $3244. So I replaced these first two bytes with the true load address plus two, in a low/high byte order, as in $46 $32.

I'm done with the disk editor at this point and used a disassembler program. The disassembler read in the load address -- my user supplied value of $3246 -- and went from there, disassembling that particular VLIR record. I used a symbolic disassembler modified for LADS 128 by James Mudgett and it works in 128 mode. Theoretically, I could supply a symbol file containing GEOS equates and it will produce some GEOS compliant source code. I haven't done this yet. Next, I used the Make-ASCII program by the authors of EBUD to convert the PRG disassembly to a SEQ petascii file.

I used EBUD to modify the source code file, add GEOS equates, fix some minor addressing problems in the ML code, add comments, etc. In short, I followed the program flow and logic and adding comments here and there, fixing up stuff, etc. This is the true guts and grits of disassembling and takes up quite a bit of time, reference to documentation, etc. At times, this process was enlightening and at times, it was quite dull and boring. (Ever try to translate codes into ASCII text that GEOS uses?)

My best friend during this disassembly process was dialog boxes, menus, icons and prompts. Why? They shed light on the calling routines. Some routines were so obscure at an initial glance that I couldn't figure them out. But when it called a dialog box, presto! I understood now what the routine was trying to do in the first place. This disassembly process goes on and on. Maurice Randall also was very patient in answering my email correspondence as I progressed through the disassembly. In case if I haven't said it, I'm saying it now... THANKS! :)

Anyway, the reason why I decided to use non-GEOS tools is because I wanted to produce an exact copy of the original code from my disassembled code. I couldn't do that with geoProgrammer as it doesn't always produce an exact copy. Maurice has fixed many problems in geoProgrammer with the release of Concept and can be downloaded from his website. I've been using Concept to assemble the patcher program that patches a user's copy of geoWrite 128 v2.1 and have full confidence in that programming package.

Upon assembling the source code, I ran a compare of the resulting code against the original PRG file that I extracted earlier in a disk editor. I used my Action Replay v6 ML monitor to do the comparisons. If the files did not match, then I went back into the disassembly and figured out why, etc., and fixed them. Once I did have a match, then I knew my disassembly was perfect and that I could add any changes to it I wanted, etc., and is what I've done with geoWrite 128 v2.2.

Having a perfect copy of the VLIR #5 to work with, I went back into my disk editor to patch it back into the VLIR application on a 5 1/4 disk. I did this by modifying the VLIR table as to include that PRG file as a VLIR record and remove the directory entry pointing to it as a PRG file. Remember the modified load address of $3246? I removed it and added the original two bytes of code, so the GEOS application would run fine within the GEOS environment.

As for serialization and the GEOS Kernal ID check, I don't want to get into this area, but suffice it to say that I used the Action Replay v6 ML monitor to remove these. You may have to remove these if the code you want to disassemble is somehow blocked by serialization or serial number checks. This is also an another reason why I decided to use non-GEOS tools. GEOS can't remove the serialization by itself, obviously, and Action Replay v6 does not run in 128 mode and may crash the computer when running in 64 mode running GEOS. The GEOS Kernal ID check is a form of copy protection and is used often

by major BSW applications.

This is largely the process I've been using to patch geoWrite 128 v2.1 to a v2.2 by modifying VLIR #0 and #5. I haven't touched other modules yet and there are 7 more modules to go. :(Hopefully I haven't missed anything and that it has been helpful and instructive. I plan to finish disassembly of geoWrite 128 v2.1 and will certainly post more disassembly notes in the future, either in C=Hacking or wherever appropriate.

Disassembly Nota Bene

The rest of the article will mainly focus on routines that I've either disassembled or revamped in the geoWrite 128 v2.2 upgrade. The labels as used are largely from BSW itself, in the Official GEOS Programmer's Reference Guide and the Hitchhiker's Guide to GEOS. The format for the article is as follows:

* The routine name is displayed, along with its actual address and VLIR module number and then the category it is under. For illustration of the admittedly unorthodox addressing scheme, let's give an example: $3a41x5. The $3a41 is the actual address, and the x5 is the VLIR module number of where it is located. (VLIR modules #0 through 8)
* A brief description of the routine follows in the function field.
* Parameters, if any, are described.
* Variables or routines that the routine will use are listed.
* Listing of any variables or flags that are set or accessed when the routine is done.
* Posting of any psuedo-registers or registers that are used by the routine and not preserved.
* The description of the routine, with a sequence of events or branches.
* Finishes up with any proposed modifications that would be suitable for future work.

The disassembly comments pertaining to VLIR #5 of geoWrite 128 are only applicable to the v2.2 version as patched, and not its original v2.1 version. Traces of the original v2.1 version still remain, notably the lbxxxx labels throughout the disassembly. The numbers following the 'lb' refers to the actual locations in the v2.1 version. Any absolute addresses that do not contain a VLIR number is presumed to be addresses for VLIR #0. Also, this is not 100% complete, there are still several gaps of which I have not been able to figure out and they are duly noted, or have been guesstimated as such.

The patcher program is now available and is being marketed by yours truly, a multinational corporation. :) Ok, so I'm no corporation nor paper tiger, but email me for more details at eyethian@juno.com about acquiring the patcher program.

There is a geos programming emailing list maintained at cbm.videocam.net.au. One need not join, but can read past archives covering various topics of geos programming. The URL for the geoProgramming:The Millennium (GTM) emailing list is: http://cbm.videocam.net.au/gtm/

That all said, there are plenty of GEOS programs, especially those BSW applications, that are ripe for disassembly for further exploration and knowledge of GEOS programming. As a bonus, these programs really do need modifications as to make them work all but seamlessly on modern day GEOS systems with hulking CMD power.

I hope that the following disassembly notes will be instructive to people wishing to get into GEOS programming or to upgrade other existing BSW applications. At any rate, enjoy.

```
setProgDevice &         - $0daax0 -             Disk Routines
setDataDevice           - $0daex0 -

Function:       Sets the disk device from which the application was loaded
from. Sets the disk device for which the datafile activities take place.

Parameters:     None.

Uses:           progDrive
                dataDrive
                toggleZP
                SetDevice

Returns:        None.

Destroys:       .A

Description:    Depending on the entry point, it checks either progDrive or
dataDrive and issues a SetDevice call to change the active disk device.
```

Proposal: This needs to be overhauled, as the disk device driven model is obsolete in modern day GEOS systems. This needs to be changed to a directory driven model, where directories rule and not the disk device. Users can change directories, subdirectories, partitions, etc. and the program needs to keep track of all of this activity and the current routines fall short.

VLIROps - $0dddx0 - Disk Routines

Function: Loads in geoWrite 128's own individual VLIR modules.

Parameters: .A containing the VLIR number for the geoWrite 128 module to load.

Uses: PVLIROps
CVLIRNum - Current geoWrite 128 VLIR module in memory
setProgDevice
RVLIRB2F - restores a geoWrite VLIR module from BackRAM
$3172 - Address location of which starting tracks and sectors for the nine individual VLIR modules can be found and is used as an index.
$2798 - ReadFile
SVLIRF2B - saves a geoWrite VLIR module to BackRAM
GotoFirstMenu
lb29dc - Error message
lb233a - Canned DB to display error message

Returns: None.

Destroys: r1, r2, r7, r10, .A, .X and .Y.

Description: First, it loads r7 with $3244 for the loading address for the VLIR modules. Next, it copies r7 into r10 for the error handler. Next, it checks the value passed in .A against CVLIRNum and if they match, then the VLIR module in question is already in memory and there is no need to load it in from the disk device, and it simply RTS's without any further action.

If the numbers do not match, then CVLIRNum will take on the number passed in .A and opens the disk device housing the application through setProgDevice. Next, it calls RVLIRB2F, and sees if a copy of the VLIR module is already located in BackRAM. If so, the routine simply restores the VLIR module by fetching it from BackRAM, and doesn't need to access the disk device at all.

If the VLIR module is not located in BackRAM, it then checks the index located at $3172 and retrieves the starting track and sector of the VLIR module and issues a ReadFile to load in the VLIR module. The maximum VLIR module size is only 4,000 bytes. Then it calls SVLIRF2B, to determine if the newly loaded in VLIR module should also be saved to BackRAM. If there is an error, a DB is generated and the VLIR module in question tries to get accessed again.

Note: There is an alternative entry point, PVLIROps, which is for the VLIR #8, the printing code. The reason is that this module loads in at $0680 as opposed to $3244 for VLIR modules #1 through 7. The entry point is located at address $0de5, or just eight bytes beyond VLIROps, and requires passing r7 the $0680 address.

Proposal: This routine can be modified as to allow retrieval of VLIR modules from expansion ram in Wheels or MP3 operating systems. Already implemented in the patcher program for Wheels users.

StashGW128 - $2bdbx0 - Setup

Function: Stashes all eight VLIR modules of geoWrite 128 v2.2 into expansion memory, and enables the ram-based VLIR code.

Parameters: a7L (Wheels flag) and $d4 (LoadOptFlag)

Uses: RAMOps - Core ram-based VLIR code
setProgDevice
obtainRec - loads in an individual VLIR module from disk
$3172 - Address location of which starting tracks and sectors for the nine individual VLIR modules can be found and is used as an index.
$2798 - ReadFile
standard - loads r7 with $3244
deviate - loads r7 with $0680
checkDisk
i_MoveData
VLIRAMOps - Replacement routine for the following routine:

VLIROps - the routine being replaced by the previous routine.
CVLIRNum - Current geoWrite 128 VLIR module in memory
toggleZP
DoRAMOp
EnterDeskTop
lb29dc - Error message
lb233a - Canned DB to display error message

Returns: None.

Destroys: r0, r1, r2, r3L, r7, r15L, .A, .X and .Y.

Description: First, it checks a7L and determines if it is running under a GEOS 128 or Wheels 128 system. If it is GEOS 128, then the routine simply jumps to $3244 of geoWrite's VLIR #1 and goes from there. Secondly, it checks LoadOptFlag to determine if the datafile was passed through the printer icon for printing. If that is the case, then the routine would jump to $3244 of geoWrite's VLIR #1. In either event, no RAM activities take place.

If the routine gets past the checks, then it stashes VLIR #1 into expansion memory because it is already sitting there in FrontRAM memory. Next, it calls setProgDevice and proceeds to stash VLIRs #2 through 7 into expansion memory. Lastly, it stashes the memory region from $0680 to $0b80 into expansion memory. Then it loads in VLIR #8, the printing code, into that region beginning at $0680. Next, it performs a SWAP of memory located at $0680 and that in expansion memory, putting the printing code into expansion memory and restoring the original code located at $0680.

Next, it fetches VLIR #5 from expansion memory and modifies the checkDisk code as to allow the DISK icon to be placed in the file requestor, even if it is displaying the disk device from which geoWrite 128 originally loaded from. Next, once the code is modified, VLIR #5 is stashed into expansion memory.

Lastly, VLIR #1 is fetched from expansion memory. Next, it replaces the code as found in VLIROps with the ram-based VLIR code located at VLIRAMOps. Finally, it jumps to $3244 in geoWrite's VLIR #1.

Proposal: None. Code may change to support ram expansion capabilities of MP3 128 systems.

VLIRAMOps - $0dddx0 - RAM Routines

Function: Loads in geoWrite 128's own individual VLIR modules via expansion RAM.

Parameters: .A containing the VLIR number for the geoWrite 128 module to load.

Uses: PVLIROps
RAMOps
CVLIRNum - Current geoWrite 128 VLIR module in memory
toggleZP
DoRAMOp
lb29dc - Error message
lb233a - Canned DB to display error message
EnterDeskTop

Returns: None.

Destroys: r0, r1, r2, r3L, r7, .A, .X and .Y.

Description: First, it loads r7 with $3244 for the loading address for the VLIR modules. Next, it checks the value passed in .A against CVLIRNum and if they match, then the VLIR module in question is already in memory and there is no need to load it in from RAM expansion, and it simply RTS's without any further action.

If the numbers do not match, then CVLIRNum will take on the number passed in .A and checks to see if it is VLIR #8, the printing module. If that is the case, then only 1,280 bytes gets fetched from expansion memory. Otherwise, it's 4,000 bytes. Next, it calculates the RAM expansion address offset of which the correct VLIR module can be found.

Lastly, when all registers are prepped, DoRAMOp does the job of fetching the VLIR module into the correct memory location. If there is an error, a DB is generated and the application aborts to deskTop.

Note: There is an alternative entry point, PVLIROps, which is for the VLIR #8, the printing code. The reason is that this module loads in at $0680 as opposed to $3244 for VLIR modules #1 through 7. The entry point is located at address $0de5, or just eight bytes beyond VLIRAMOps, and requires passing r7

the $0680 address.

Proposal: This routine replaces the disk-based VLIR routines as found in geoWrite 128 v2.1. This routine may need to be changed to support the MP3 128 platform.

checkDrives - $2b0dx0 - Disk Routines

Function: Checks available drives on a user's system.

Parameters: r0L via deskTop with LoadOptFlag
r2 via deskTop pointing to diskname of the disk containing the datafile.
r3 via deskTop pointing to datafile's filename.

Uses: DrACurDkNm
DrBCurDkNm
DrCCurDkNm
DrDCurDkNm
setDataDevice
setProgDevice
$27b0 - FindFile
curDrive
numDrives
toggleZP
prepDlgBox
enterDeskTop

Returns: progDrive - The drive geoWrite was loaded from.
dataDrive - The disk device that houses the datafile.

Destroys: r6, .A, .X and .Y

Description: First, it checks curDrive and stores the value there into progDrive. Next, it stores the same value into dataDrive. It checks then the LoadOptFlag to determine if a datafile was clicked on. If so, it checks the name of disk A and compares it against the name of the disk that has the datafile. If there is a match, then the datafile is on drive A or C. Next, a value of eight is stored into dataDrive.

If they are different, then the routine checks for additional drives. If there are no additional drives, then a dialog box is fetched, warning the user that a datafile and geoWrite must be on the same disk on single drive systems. After that dialog box is over, geoWrite quits to deskTop. If there is an additional drive, then geoWrite will put a value of nine into dataDrive.

Proposal: This code can be modified to search drives A-D in sequence. This way, a user can click on a datafile anywhere in his/her system and geoWrite 128 can boot up correctly. Secondly, it should use FindFile to nearly pinpoint the proper disk containing the datafile because disknames can be identical. Already implemented in the patcher program.

checkVer - $2b75x0 - Misc. Routines

Function: Checks the current GEOS version.

Parameters: None.

Uses: Version
c128Flag
DoDlgBox
enterDesktop

Returns: None.

Destroys: r0 and .A

Description: First checks to see if it is running in GEOS v2.0 or higher systems. Next, it checks to see if it is running on the 128 version of GEOS. If both conditions are not met, then a dialog box is printed to that effect and geoWrite 128 aborts to deskTop. Otherwise, the routine RTS's w/o further action.

Proposal: The routine can be modified to check for Wheels or MP3 systems and designate a variable for later routines to rely upon. Already implemented in the patcher program, where it designates a7L as the Wheels flag.

toggleZP - $251bx0 - Housekeeping

Function: Toggles the state of all zp variables, thereby preserving two zp spaces, one for the actual application usage and the other for stock GEOS/Kernal usage.

Parameters: None.

Uses: None.

Returns: None.

Destroys: None. All registers are preserved via the stack.

Description: It's a toggle routine. When called, it performs a swap of zero page space located at $80 to $fb towards a buffer located in $2e22, and the buffer contents are similarly swapped back. This way, geoWrite 128 can use that zp region freely, and swap it out whenever calling GEOS Kernal routines that rely on this area, and when these routines are done, the toggle routine is called again to swap back in those values for use in the application. This is the most heavily used routine in geoWrite 128.

Proposal: While this routine is nice as it allows the application more zp space, this is unnecessary routine and should be eliminated. It slows down the application somewhat, as this must be called twice whenever a GEOS Kernal call is used.

prepDlgBox - $2314x0 - Dialog Boxes

Function: Preps the pointers for the actual DoDlgBox function call.

Parameters: .A has low byte and .Y has high byte pointing to the dialog box text.

Uses: DoDlgBox
$2538 - Does something to the VDC

Returns: Carry flag is set.

Destroys: r5, r14, .A, .X and .Y

Description: The routine uses a `canned' dialog box with preset icons, placement and size. The only thing that is controllable is the text. The pointer passed on in .A and .Y registers points to the text. Commonly used to create error dialog boxes with an OK icon.

Proposal: None.

layoutScreen - $32cex1 - Appearance

Function: Draws up the main menu bar and the overall screen layout for geoWrite 128.

Parameters: None.

Uses: DoMenu
i_GraphicsString

Returns: Carry flag is set.

Destroys: r0, .A

Description: The routine draws the screen layout using various GraphicsString parameters. Additionally, it builds the main menu bar at the top, with its table located at $0bbcx0.

Proposal: This may need changing, especially if new menu items are added or old ones deleted and maybe the screen layout should be changed or left as is.

InitGW128 - $3c70x1 - Startup

Function: Initializes geoWrite 128 with variables, flags and vectors.

Parameters: None.

Uses: toggleZP
closeRecordFile
dispBufferOn
RecoverVector
checkSerNbr

Returns: None.

Destroys: .A

Description: The routine sets the following locations to zero: $41e4-e5, $0200, $021a, $2dfa, $db, $de, $e1, and $f7. It also closes geoWrite 128's own VLIR record, and as well as activates the software mouse as sprite zero. It sets the dispBufferOn flag as to allow only writes to foreground and sets the RecoverVector to point at $2199. It sets $f1 to have a value of $c0 and sets $023a & $023c to have a value of $ff.

Proposal: This may need changing, when new variables or flags need be set.

CheckPtrDrv - $33e7x1 - Startup

Function: Checks the status of the printer driver and loads it in, if necessary.

Parameters: None.

Uses: loadPtrDrv
getDimensions
setProgDevice
maxPtrHgt ($2dfb)

Returns: None.

Destroys: a9L, .A, .X

Description: The routine sets the maximum printer height ($02f0) in card rows to maxPtrHgt. Then it loads in the printer driver, and if there is an error, no harm is done as the maximum printer height is already established. But, if a printer driver is successfully loaded in, then a call to getDimensions will place the maximum printer height in card rows at maxPtrHgt.

Proposal: Why not just call getDimensions when the printer driver is always loaded into memory in GEOS 128 configuration?

InitGW128 - $325cx5 - Dialog Boxes

Function: Prints the copyright message and calls the infamous Create, Open or Quit DB.

Parameters: None.

Uses: $1baf - Turns off text cursor.
DrawStripes
i_GraphicsString
printCopyrightMsg
$2538 - issues a dialog box
createOpenQuitDB - DB table

Returns: None.

Destroys: all registers.

Description: It turns off the text cursor, draws the striped pattern you see in the upper right corner of the screen, clears the screen with a default pattern, prints out the copyright message, and issues the infamous create/open/quit DB. The routine will then branch to routines that handle creation of datafiles, opening of datafiles or quitting the application.

Proposal: None I can think of right now. Maybe abolish this dialog box in favor of allowing the user to boot geoWrite 128 to a blank screen and have him/her to select from a menu an appropriate action to undertake.

printCopyrightMsg - $3375x5 - Text

Function: Prints the copyright message.

Parameters: None, but can only be used once.

Uses: crflag
$1f05 - calls system font
currentMode
i_GraphicsString
i_PutString

Returns: None.

Destroys: r1, .A and possibly others.

Description: Sets the crflag as to reflect that the copyright message has been printed. That's why this routine can only be used once. It calls the system font, and then uses inline routines to draw and print the copyright message.

Proposal: No change. Unless someone patching this copy wants to add their own message.

createDocRoutine - $33ddx5 - Disk Routines

Function: Creates a new geoWrite v2.1 datafile.

Parameters: none.

Uses: queryFilename
nameBuffer
$27b0 - FindFile
lb3f67 - File exists text prompt
$2314 - issues a DB
createNewFile
$de - currently unknown
$d3 - currently unknown
currentMode
TitlePage
HeaderHeight
FooterHeight
FirstPage
PageHeight
CPageHeight
PageWidth
gPFlag
$2393 - Something to do with fonts
SetScrollSprite
printDataName
lb404a - creating file error text string
$233a - issues a DB
InitGW128

Returns: None.

Destroys: r6, .A, .X and possibly others.

Description: First turns on the icons related to drive activity and then calls the DB that prompts the user to enter the filename. If the user cancels or no filename was inputted, the routine then goes back to the main DB in InitGW128. r6 is then loaded with the inputted filename and the FindFile function is called. If a file exists on disk, the routine informs the user with a DB and prompts the user for a new filename. It then calls createNewFile and sets up variables relating to page length and width, margins, headers and footers, etc. Next, it sets up the fonts and the scrolling sprite. Last, it will print the datafile's filename in the upper right corner of the screen.

Proposal: One thing could be changed; instead of requiring the user to make a new filename upon notice that a file already exists, the user could override it and use the same filename, but get rid of the old one. This could be a `TEMP' file concept. If a new version of geoWrite is written with a new version of a datafile, then this routine would need be revamped accordingly.

Setup4DB - $3449x5 - Disk Routines

Function: Preps the dialog boxes with the appropriate drive icons and the DISK icon.

Parameters: None.

Uses: nameBuffer
setDiskIcon
checkDisk
setDataDevice
readDiskName
iconLTable
iconHTable
driveType
curDrive
onLTable
onHTable
offLTable
offHTable

Returns: None.

Destroys: r1, r7H, r0, r5, .A, .X & .Y

Description: First, it delimits the nameBuffer. It then checks for a ramdisk and application disk in order to turn on/off the DISK icon. It then reads in the disk name so that it will appear in the requestor DB. Next, it checks all available drives and activates the drive icons accordingly. Last, it loads in r5 with nameBuffer.

Proposal: No change.

fileRequestor - $34b2x5 - Dialog Boxes

Function: Calls up the file requestor box w/ 4 drive support.

Parameters: None.

Uses: setup4DB
permName - (Write Image)
restoreDBTable
dBTable - DB table w/ DBGETFILES
$2538 - Issues a DB
InitGW128
curType
lb3f45 - pointer to text (Insert New Disk)
$2314 - Issues a DB
$27ad - OpenDisk
openServiceRoutine
dataDrive
setDataDevice
toggleZP

Returns: None.

Destroys: r7L, r10, r0L, .A, .Y

Description: When called, the routine first preps the drives via the setup4DB routine. Since the dBTable would be shared by other requestors, some modifying code was used. The routine modifies the dBTable w/ restoreDBTable data. It sets up DBGETFILES to search for only APPL_DATA files containing the permanent name string of `Write Image'. Finally, the DB is issued, complete w/ 4 drive support.

If the user cancelled, then it aborts back to InitGW128. If the user opened a file, then openServiceRoutine is run. If the user clicked upon the disk selection, it checks to see if non-removable media is present. If it is removable media, then an another DB is issued to prompt the user to enter the new disk. Otherwise, it displays the file requestor again w/ new contents of the newly selected partition. If a drive icon is clicked, it will issue a setDataDevice command and OpenDisk the new drive and repeats the file requestor w/ new contents of the newly selected drive.

Proposal: No change.

openServiceRoutine - $364dx5 - Disk Routines

Function: Opens a geoWrite v2.1 datafile.

Parameters: None.

Uses: setDataDevice
nameBuffer
InitGW128
$27b0 - FindFile
TSDFHdr
dirEntryBuf
lb35fc - pointer to DB data (File Write Protected)
$2538 - Issues a DB
PageWidth
CheckDFVer
lb3f86 - text pointer (version higher than v2.1)
$2314 - Issues a DB
convertDataFile
lb356e - extracts global variables stored in a datafile's file header
$260c - r0 points to filename
OpenRecordFile
toggleZP
AdjPageWidths

windowBottom
$d3 - currently unknown
$de - currently unknown
SetScrollSprite
printDataName
lb403d - text pointer (opening file error)
$233a - Issues a DB

Returns: None.

Destroys: r5, r6, r0L, .A, .X & .Y

Description: First, it calls setDataDevice, and checks nameBuffer to see if a filename was selected. If no name is selected, it quits to InitGW128, otherwise, it will find it by FindFile. It then stores the T/S for the datafile's fileheader, and checks to see if it is write-protected. Next, it stores in a max width of 639 into PageWidth. Next, it checks the datafile's version and if necessary, converts it to a v2.1 format. Next, it gets global variables stored in the datafile's fileheader and stashes it into zero page. Last, it finally opens the datafile with OpenRecordFile. It will also read in geoPublish data and adjust page widths. It stores a $00 in $d3 and $de and as well as a $c7 (scanline 199) in windowBottom, Last, it will set the scrolling sprite and prints the datafile's filename on the upper right corner of the screen.

Proposal: Changes may be necessary if new formats are to be supported in addition to regular geoWrite v2.1 datafiles.

checkDisk - $371bx5 - Disk Routines

Function: Checks to see if the data drive is a ramdisk or holds the application and sets the DISK icon accordingly.

Parameters: None.

Uses: a7L
dataDrive
progDrive
driveType
cableType

Returns: Carry clear indicating the existence of a ramdisk or that the application is on that same disk as designated as a data device. Carry set means that the DISK icon can be placed in a requestor DB.

Destroys: .A, .Y

Description: First, it checks to see if the application is on the same disk as being designated as a data device housing the datafile. If that is the case, then the DISK icon cannot be placed. Next, it checks the appropriate driveType entry to check the existence of a ramdisk. If a 41/71 ramdisk is being used, then the carry flag is cleared as to prevent the DISK icon. If a 81 ramdisk is being used, then it checks the Wheels flag at a7L to determine if it's a 81 RL ramdisk as shown in cableType. If that is the case, then the carry flag is set as to allow the DISK icon, otherwise it is cleared as to prevent the DISK icon because it's just a regular 81 ramdisk. The carry flag is set if a native ramdisk is being used, as to allow the DISK icon to appear.

Proposal: If geoWrite 128 can fully load itself into RAM, then the DISK icon can be placed in the requestor DB even when the disk device houses both the application and the datafiles. As it stands, the DISK icon must be removed because the user might select a different disk, partition or subdirectory and the application then can't find its own VLIR modules. Instant crash.

UPDATE I have added ram routines and geoWrite 128 is now 100% RAM resident, and therefore, the DISK icon can be displayed in this case. But there is one major caveat, among others, in using this approach. The fonts are keyed to the disk device from which geoWrite 128 was loaded from. If the user changes the disks or partitions or subdirectories from that same disk device, then geoWrite 128 would not be able to find the font data and may lead to unpredictable results. The same goes for Text Scraps and other features that are tied to the disk device from which geoWrite 128 originally loaded from. In the future, I would have to work on font routines and other routines as necessary as to eliminate this dependence on the original disk device.

readDiskName - $3739x5 - Disk Routines

Function: Reads in a disk name housing the current datafile.

Parameters: None.

Uses: DrACurDkNm
DrBCurDkNm
DrCCurDkNm
DrDCurDkNm
diskName
dataDrive

Returns: None.

Destroys: r0, .A, .Y

Description: Depending on the value of dataDrive, r0 is loaded with the appropriate current diskname and its contents are transferred to diskName. This way, the file requestor can display the disk name along with other info.

Proposal: No change.

recoverFile - $3781x5 - Disk Routines

Function: It recovers a file.

Parameters: None.

Uses: GotoFirstMenu
$dd - some kind of unidentified flag
i_MoveData
CNameBuffer
NameBuffer
openServiceRoutine
lb4005 - text pointer (Cannot recover)
$2314 - Issues a DB

Returns: None.

Destroys: .A, .Y

Description: Calls GotoFirstMenu to close its menu selection, then tries to recover the file by copying the current filename into the filename buffer and calling openServiceRoutine. It does check the flag at $dd before determining whether a file could be recovered. If it can't be recovered, a DB is issued to that effect.

Proposal: No change.

RenamFile - $379bx5 - Disk Routines

Function: It renames a file.

Parameters: None.

Uses: GotoFirstMenu
queryFilename
CNameBuffer
NameBuffer
$27b0 - FindFile
toggleZP
RenameFile
printDataName
SetScrollSprite
lb40a7 - text pointer (File Exists error)
$2314 - Issues a DB
$260c - r0 points to NameBuffer

Returns: None.

Destroys: r6, r0, .A, .X, .Y

Description: First calls GotoFirstMenu before calling queryFilename. The new name is then put into CNameBuffer and calls the RenameFile routine. Of course, if a file exists, then the DB pops up, reporting the error. Last, it prints the new filename onto the upper right corner of the screen and sets the scrolling sprite.

Proposal: No change.

queryFilename - $37dbx5 - Dialog Boxes

Function: Prompts the user for a filename for a v2.1 datafile.

Parameters: .A must pass either a zero or a $12 to turn off/on the drive icons and the system DISK icon from the DB.

Uses: renTable
setup4DB
NameBuffer
lb3f45 - text pointer (Insert New Disk)
$2314 - Issues a DB
replaceDBTable
dBTable
qDBTable - DB table data
$2538 - Issues a DB
dataDrive
setDataDevice
$27ad - OpenDisk
curType

Returns: .A containing the value of r0L.

Destroys: r0L, .A, .X & .Y

Description: First, it shuts off/on the drive icons and the DISK icon before calling setup4DB to prep the DB with appropriate icons. Next, it modifies the DB table to replace the DBGETFILES with DBGETSTRING, as this DB table is also shared by the file requestor. If a drive icon was clicked upon, the drive gets accessed and the DB is reissued with the updated icon data. If a DISK icon was selected, it will prompt the user to insert a new disk, or skips that process if it's on non-removable media. Last, upon exiting, it loads .A with r0L so that the calling routine will know what the user selected.

Proposal: No change.

createNewFile - $3839x5 - Disk Routines

Function: Creates a new geoWrite v2.1 datafile.

Parameters: None.

Uses: CPageHeight
NameBuffer
lb38c4 - (word) Page Height stored in a datafile's fileheader
dFileHeader
toggleZP
SaveFile
$27b0 - FindFile
CheckDFVer
dirEntryBuf
TSDFHdr
$260c - r0 points to filename
OpenRecordFile
$27a7 - AppendRecord
$27aa - UpdateRecord
PointRecord

Returns: None.

Destroys: r10L, r9, r6, .A, .X & .Y

Description: First, it stores the current page height into the global variables as hidden in the datafile's fileheader. Next, it points the first two bytes of the datafile's header (dFileHeader) to the filename and calls SaveFile. Next, it calls FindFile to load in the newly created datafile's fileheader into memory and preserves its t/s pointers. It also checks its version identifier. Next, it opens the datafile and creates 127 blank VLIR records and updates it and then finally points to VLIR #0 for further handling.

Proposal: Changes may be necessary to support a newer format or support other file formats.

convertDataFile - $393ex5 - Data Handling

Function: Converts a datafile to a v2.1 format.

Parameters: .Y contains $31 or higher to correspond with ascii values of 1 thru 9. .A contains $32 or lower to correspond to ascii values of 2 through 0, i.e., .A is the `2' in `v2.1' and the .Y is the `1' in the `v2.1'.

Uses: lb3fbf - points to the version string of `v2.x' where x = 1 or higher

VerFlag
convertFileDBTable
$2538 - Issues a DB
toggleZP
sysDBData
fileHeader
lb38bd - global variables controlling document
lb356e - extracts these global variables from the file header
TSDFHdr
$27b6 - PutBlock
$260c - r0 points to filename
OpenRecordFile
PointRecord
lb3978 - modifies a VLIR record of the datafile
NextRecord
$279e - Closes the VLIR datafile
lb3fc1 - text pointer (Converting File Error)
$233a - Issues a DB

Returns: .A to indicate error status ($00 = no error)

Destroys: r4, r1, .A, .X & .Y

Description: First, it modifies the file header of the datafile as to make it to show v2.1. Depending on the values passed, it will set VerFlag as to control further file conversion to a v2.1 format. Next, it issues a DB informing the user that it's converting the datafile to a v2.1 and asks permission. Then it modifies the fileheader of the datafile as to incorporate new global variables. With the fileheader modified, a PutBlock call is issued.

Finally, it will read in all used VLIR records of the datafile and convert them to a v2.1 format. Next, it will read in the header and footer VLIR records and modify them as well. Last, it will then close the datafile.

Proposal: Changes may be necessary to support a newer format or support other file formats.

lb3978 - $39f0x5 - Data Handling

Function: Converts an individual VLIR record of a datafile to a v2.1 format.

Parameters: The VLIR record must have been already opened.

Uses: fileData
VerFlag
toggleZP
ReadRecord
lb39da - fixes v1.x ruler escapes to conform to the v2.1 standard
lb3fbf - the V2.X identifier string where X is accessed
lb3a09 - fixes the width of a v2.0 page to a v2.1 page
WriteRecord

Returns: None.

Destroys: r2, r7, .A, .X & .Y

Description: It will first read in a VLIR record of a datafile, and depending on its original version (set by VerFlag), it will modify ruler escapes as to make the current record conform to v2.1 specifications. When all of these modifications are done, the record is written back w/ WriteRecord.

Proposal: Changes may be necessary to support a newer format or support other file formats.

lb39da - $3a52x5 - Data Handling

Function: Converts an individual VLIR record of a datafile from a v1.x to a v2.1 format.

Parameters: The VLIR record must have been already opened and read w/ ReadRecord.

Uses: fileData

Returns: None.

Destroys: .A & .Y

Description: It merely moves the first 20 bytes of the first ruler escape on a VLIR record back by seven bytes and zeroes out some parts of the ruler escape.

Proposal: Changes may be necessary to support a newer format or support other file formats.

lb3a09 - $3a81x5 - Data Handling

Function: Converts an individual VLIR record of a datafile from a v2.0 to a v2.1 format.

Parameters: The VLIR record must have been already opened and read w/ ReadRecord.

Uses: $d0 - word pointer to last byte of individual VLIR record of a datafile
$25a1 - points r15 to start of fileData
toggleZP
$26d7 - compares r15 against $d0
$25e1 - increments r15
$25db - increments r15 three times
$25f7 - skips ruler escapes pointed to by r15

Returns: None.

Destroys: r7, r15, r1, .A & .Y

Description: The v2.0 page has a width of 1.2 to 7.2 inches, and the v2.1 page has a width of 0.2 to 8.2 inches, and this routine searches for every ruler escape and converts them to a v2.1 format.

Proposal: Changes may be necessary to support a newer format or support other file formats.

printDataName - $3af1x5 - Appearance

Function: Prints the datafile's filename in the upper right corner of the screen.

Parameters: nameBuffer must have a filename.

Uses: nameBuffer
CNameBuffer
i_MoveData
$1fba - use system font
currentMode
DrawStripes
PutChar
PutString
rightMargin
GetCharWidth

Returns: None.

Destroys: r0, r1, r11, r13L, .A & .Y, r2L is specially preserved.

Description: It moves the contents at nameBuffer to CNameBuffer and then calculates the width of the filename, ensuring that it does not exceed its maximum width. Next, it will print a trailing space, then the filename, and then a leading space, in that little striped box in the upper right corner of the screen.

Proposal: No change.

runDA - $3bb1x5 - Desk Accessories

Function: Allows a DA to be run.

Parameters: .A passes the index number of the DA in an internal table.

Uses: GotoFirstMenu
$d3 - I think this variable holds the current VLIR # of a datafile
lb406f - text pointer (Cannot run DA)
$2314 - Issues a DB
$07c5 - Inverts a rectangle and preserves zp space
$0d61 - Closes the current datafile
saveTScrap
$21ba - saves the portion of the screen
$01 - I/O port

$d017 - sprite horizontal expand register
$4c95 - Currently unknown word variable
setProgDevice
toggleZP
GetFile
$21bf - restores the portion of the screen
i_MoveData
lb3ecd - text pointer (Not enough disk space)
lb4058 - text pointer (Running DA error)
$233a - Issues a DB
$22e3 - Sets up the text cursor
loadTScrap
$0d6c - reloads the VLIR datafile and points to the last accessed record
$851e - default screen color
VIDEO_MATRIX
FillRam
$2555 - Not sure yet what this routine does
$1512 - Displays the characters onscreen
$2575 - The opposite of $2555
SetScrollSprite
$1bbb - turns on the text prompt and draws a couple of rectangles

Returns: None.

Destroys: r6, r15L, r0, r2L, r1, r10L, .A, .X & .Y, and since a DA is run, pretty much everything else has been hosed, except that geoWrite 128 tries to preserve as much as possible.

Description: After calling GotoFirstMenu, it checks to see if it's in header/footer mode and if so, the DA can't be run. Next, it preserves much of zero page and inverts a rectangle twice onscreen. It then closes the current VLIR datafile, saves any Text Scraps, sprite data/expansion registers and the first 24 scanlines of the screen. Finally, it calls and runs the DA. When the DA is done, geoWrite then will restore sprite data/expansion registers, the first 24 scan lines of the screen, color and video data and much of the zero page area. Then it sets up the text cursor, loads in any Text Scrap, and reloads the VLIR datafile and points to its current VLIR record. Next, it will begin displaying the contents of the datafile, enable the text cursor and draws a couple of rectangles, and returns control to MainLoop.

Proposal: The I/O calls seem redundant for I/O is always banked in and color data is being restored as well, which is puzzling as geoWrite 128 does not support color. Other than that, no changes.

saveTScrap - $3c97x5 - Disk Routines

Function: Saves a text scrap.

Parameters: None.

Uses: TSFlag
TSiZe
lb3dff - load address of text scrap
lb3e01 - size of text scrap
setProgDevice
lb40c1 - pointer to filename (Text Scrap)
toggleZP
DeleteFile
lb3db8 - pointer to filename for the SaveFile call and also doubles as its fileheader
SaveFile

Returns: TSFlag holds a zero value to indicate that a Text Scrap is saved, otherwise $ff if something's wrong.

Destroys: r0, r9, r10L, .A & .X

Description: It checks TSFlag and TSiZe to determine existence of a text scrap, and if so, then attempts to save it. It modifies the fileheader for the text scrap as to include an appropriate load address ($41e4), the ending address, etc. Next, it saves them to the same disk/partition as where the geoWrite 128 application resides. It will delete a prior text scrap if one existed.

Proposal: The area, $41e4-$4310, seems too small of a buffer for a text scrap. Maybe this buffer should be enlarged to accommodate large text scraps. Also, this may need to be changed to accommodate future versions or simply accommodate a text album file.

quitGeoWrite - $3cb1x5 - Disk Routines

Function: Quits the geoWrite 128 application and returns to deskTop.

Parameters: None.

Uses: SaveTScrap
setProgDevice
toggleZP
EnterDeskTop

Returns: None.

Destroys: None; the deskTop now takes control.

Description: First, it saves the current Text Scrap. Then it returns to the disk/partition that it was originally booted from, restored zero page space and then quit, resuming control to the deskTop.

Proposal: No change.

SetScrollSprite - $3cbdx5 - User Interface

Function: Enables the scrolling sprite that one uses to scroll the document onscreen, in the little box in the middle top of the screen.

Parameters: None.

Uses: $01 - i/o port
$d02d - Sprite 6 color register
i_FillRam
i_MoveData
lb3c73 - sprite data
$d01d - expand sprite 6 horizontally

Returns: None.

Destroys: .A and $01 is preserved.

Description: It enables I/O, colors sprite #6 black and expands it horizontally, and defines its bitmap.

Proposal: No change. But is the I/O activity redundant because I/O is mapped in already in a GEOS 128 configuration?

AdjPageWidths - $3cf6x5 - Data Handling

Function: Reads in imprinted geoPublish data and adjusts page widths.

Parameters: None.

Uses: gPFlag
PointRecord
toggleZP
ReadRecord
$2c1d - Page Width High Byte
$2be0 - Page Width Low Byte
pageWidth

Returns: None.

Destroys: r2, r7, .A, .X & .Y

Description: It will check VLIR #63 of a geoWrite v2.1 datafile and determine whether geoPublish has stashed its internal data there. If so, then gPFlag is set, and geoWrite 128 will read in page widths and compensate page widths.

Proposal: Maybe give the user a DB giving an option to remove geoPublish data or leaving it alone. If a user wants to remove geoPublish data, then the document would have to be reformatted.

printInfoBox - $3d4dx5 - Dialog Boxes

Function: Prints the copyright message and info box.

Parameters: None.

Uses: GotoFirstMenu
lb3cdf - DB table pointer
infoBoxText - text pointer to copyright message, etc.

$2538 - Issues a DB

Returns: value in r0L.

Destroys: .A & .X

Description: It issues a DB stating the copyright message, its authors, etc.

Proposal: Maybe add in the patcher info here.

loadTScrap - $3d5fx5 - Disk Routines

Function: Loads in a text scrap.

Parameters: None.

Uses: setProgDevice
lb40c1 - points to filename
$27b0 - FindFile
CheckDFVer
lb40cd - text pointer (text scrap beyond v2.1)
$2314 - Issues a DB
$2625 - r4 points to $8000
$27b3 - GetBlock
lb40e7 - text pointer (reading Text Scrap error)
$233a - Issues a DB
TSiZe
TSFlag

Returns: TSiZe and TSFlag will have values reflecting the presence of a Text Scrap.

Destroys: r1, r5, r6, .A, .Y & .X

Description: It will load in a text scrap from the same disk/partition that geoWrite 128 was launched from. Next, it will check its version, and then gets info from the first datablock of the text scrap and copies down its size, sets up the flags as appropriate, where TSiZe and TSFlag will have non-zero values if a text scrap exists. It doesn't actually load in a text scrap yet, save for its first datablock.

Proposal: Changes may be necessary to support a new format or to support text albums.

CheckDFVer - $3eb9x5 - Disk Routines

Function: Checks a datafile's version against a v2.1 string identifier.

Parameters: r5 having the direntry of the datafile.

Uses: toggleZP
GetFHdrInfo
fileHeader

Returns: N and Z flags are set on whether the datafile equals v2.1

Destroys: r5, r9, .A, .Y & .X

Description: It simply gets the datafile's fileheader and compares its version string against the standard, v2.1, and sets flags as appropriate.

Proposal: No change.

....
....
..
.

::

Masters Class: xFLI Fixing

by Russell Reed <rreed@egypt.org>,
Robin Harbron <macbeth@tbaytel.net>,
S. Judd <sjudd@ffd2.com>

Last issue we covered the issues involved in NTSC/PAL fixing -- 63 cycles/line versus 65 cycles/line, total cycles per frame, etc. -- and fixed up some raster routines in a demo. One common raster routine that is usually quite easy to fix is FLI, and in this article we'll cover fixing FLI and IFLI

routines.

This article is more "interactive" than the previous article. The .zip file included in this issue contains an FLI picture (debris.panic), an IFLI picture (underwater-pal and underwater-ntsc), and some FLI display code. The FLI display code was downloaded from funet; "Underwater World", by Katon/Lepsi De/Arise, is an entry from the SymMek99 graphics competition; I'm not quite sure which competition the Debris picture is from.

Try loading and running one of the pictures, and observe what happens. That's typical of a lot, though not all, FLI-type routines.

Background

FLI is discussed in detail in C=Hacking issue #4, among other places. Briefly, FLI is a software graphics mode that gives greater flexibiliby in color selection than standard multicolor mode. FLI works by changing VIC registers to load new color information on every scan line. It uses exactly 23 cycles/line in PAL mode, and 25 cycles/line on an NTSC machine. IFLI is similar, but "interlaces" two FLI pictures, to gain even more apparent resolution and colors, at the cost of a flickering display.

Full screen FLIs are pretty standard. They have color data from $4000 to about $4FE8, graphics data from $6000 to $7F40, and more color data at either $3C00-$3FE8 or $8000-$83E8. IFLIs are not nearly so standard; you'll generally have to examine a routine to see where the graphics data and the $D800 color nybbles are stored. IFLIs can add a further wrinkle over FLI by updating the $D800 color RAM on every frame, in the vertical borders, which may require more cycles than an NTSC machine has to spare.

An FLI display routine is really quite straightforward, and looks something like

```
        initial delay
loop:   LDA xxxx
        STA $D011
        LDA xxxx
        STA $D018
        INX
        CPX #xx
        BNE :loop
```

in PAL (this article will focus on fixing PAL code, but going the other way is of course straightforward). The STA $D011 forces a badline; the STA $D018 selects a new line of color data. Sometimes this loop will be unrolled, and sometimes the initial delay can be very strange. It is this code that needs to be either modified or replaced.

Displaying a picture

The easiest way to fix a picture is to simply replace the display code. The FLI picutre "Debris" has the color data at $3C00, which is more common. Now, since FLI format is pretty standard, if you have some NTSC FLI code, it'll work on this pic. Escape the picture by either pressing the stop/restore combination, or a reset button if you have one. In the zip you'll also find "FLIview NTSC". Load and run this now, and you'll see the picture as it's supposed to look (almost -- this code isn't quite perfect; we'll get it better when we fix the code with the picture).

The next step is to save the picture in a way that can be distributed and make you famous as an 3li+3 c0de f1x3r. While the picture is still in memory, fire up a machine language monitor (most any will do, the FLI data is an an area rarely used by monitors). The easiest thing to do is to save memory from $0801 to $7F40, and compress it using ABCrunch, pucrunch, etc. -- although there are a lot of unused bytes between the display code at $0810 and the start of FLI data at $3B00, these get compressed down to just a few bytes.

Another option is to use a linker. Save memory from $3B00 to $7F40 and name it something that reminds you it's an FLI pic -- you can load this file with most any of the FLI viewers out there, and you can load it into most of the FLI editors. Copy both "FLIview" and the saved-off FLI data to a work disk. Then boot up Sledgehammer, available at most FTP sites. Sledgehammer is a compacting linker (it actually uses run-length encoding, a.k.a. char-packing). It takes a number of program and data files and "links" them together into a single file that you can load and run. Once Sledgehammer is loaded, run it, put in your work disk, and press F1 to start. You'll then see a list of the files on the disk. Use the cursor keys to move and the return key to select "FLIview" and the FLI data. Then press 'C'. Sledgehammer then asks if you need to change load addresses. You can ignore this for now, so push 'N'.

Next we see a screen asking for three pieces of information. For the name, "DEBRIS.SRF" (for self-running FLI) works. If you listed "FLIview" you saw that it did an SYS 2064 to start. 2064 is equal to $0810 in hex, so put 0810 for start, and then 37 for the $01 value. From there Slegehammer takes over, loading the files and compacting. Press space when it prompts to save. Once it's done, it resets. You can now load and run "DEBRIS.SRF", and you should see the picture. If not, read back through the article and check your steps carefully.

Okay, we've successfully made this picture work on NTSC computers, but this technique won't work in most cases (i.e. stuff besides standalone pictures). Next we're going to go back to the picture and see how to change the PAL code so it works on our computers.

Fixing the PAL code

The first thing we need to do before we fix the code is find out where it is in memory and how to start it. Also, most demos, self-running pictures, etc. are compressed, and we can't work with them until they're uncompressed. How you approach this depends on whether or not you have a freezer cartridge.

If you don't own a freezer, then fire up a monitor that sits fairly high in memory, e.g. $C000; we're going to look at the decompression code to see how to start the FLI code up. If you list the program, you'll see it does a SYS 2061, which is hex $080D. Start disassembling at $080D. You'll notice it sets border and background to black (stores color code 0 in $D020 and $D021), then calls $E544 which clears the screen. Next we have a loop to transfer a message to the top screen line - $0400. Then another loop to transfer the decompression code from $0858 to $04FF. Finally, a third loop to transfer the compressed program to high memory and a jump to $050B. This is pretty typical of most decompression code; some may skip the text/message line and/or transferring the program to high memory. I usually replace the clear screen call with three 'NOP' instructions. You may also want to replace the store to $0286, which changes the working character color to black.

Now into the decompression code we go. Start disassembling at $0858. There will be a few garbage instructions at the start. Scroll down a page or so until you see the sequence:

```
LDA #$37
STA $01
CLI
JMP $1000
```

This is where the decompression routine transfers control to the program. The instructions may vary a little on other files, but will usually be pretty similar. From this, we can tell that the program will start at $1000. We can also regain control here after decompression. Replace JMP $1000 with JMP $A7AE. Sometimes an RTS will work, and if you have a cartridge based monitor you can use a BRK. Now return to BASIC and type "PRINT 3". Some monitors disturb some floating point work areas; that line will clear this up. It may print something other than 3, but if you try it again you'll notice it's back to normal. Now run the program. After the usual depack effects, you'll notice you have a READY prompt. Reload the monitor, as the decompression probably overwrote it. Note: sometimes you won't be able to get back to BASIC, but since you know the start address, you can load the program and once it's running, use a reset button, then load your monitor.

If you have a freezer, then you can simply load and run the picture, and freeze it after it finishes decompressing.

Enter your monitor again and disassemble at $1000 -- let's study this code to see what's involved in displaying an FLI picture. The first thing we want to look at is down at $1035. The interrupt vector at $0314 is changed so that interrupts execute the code at $1043; sometimes the $FFFE vector is used instead. Now look at $1043; the part we're interested in is at $1060. Here is the main FLI display loop, as described above, that changes the vertical scroll register ($D011) and the screen display pointer ($D018), the two registers involved in FLI. If you add the cycles the instructions use, the loop takes 23 cycles. We need to extend it to 25. The easiest way here is to add a NOP at $1071. Change that part to be:

```
INX
CPX #...
NOP
BNE $1062
RTS
```

Now jump to $1000 to re-run the program. You'll notice you can see the picture, but it doesn't quite look right; the problem is in the initial delay, before the main display loop:

```
  LDY #xx
  DEY
  BNE *-2
```

Go back in and change the value at $105C until the picture looks right. You can hit stop/restore to exit the picture, then restart your monitor. Changing a value like that is a dirty way to fix the picture, but is also usually the easiest. (By the way, the value at $105C which works is $0B).

Now save your code off -- it starts at $1000 and ends at $12E0; there's tables at $1100 and $1200 as you'll notice in the FLI loop -- and crunch it down. That's all there is to it.

Now it's your turn, to fix "Underwater". This is an IFLI picture, but the process is not much different than outlined above. You need to locate the interrupt routine, and fix up the main display loop and the initial delay; unlike some IFLIs this one is pretty easy to fix, and you can always peek at the fixed code if you really need a hint.

Things that can go wrong

Load up "debris" again. This time, instead of adding a NOP to the end of the loop, add it to the _beginning_ of the loop (move the loop down one byte and add a NOP), and change the BNE to branch to this NOP. Now fix up the initial delay, and -- it doesn't work! And it doesn't really matter what value you use in the LDY #xx. This highlights the fact that FLI involves somewhat delicate timing; the problem here is that DEY BNE *-2 takes five cycles, which is too "coarse". Some NOPs are needed instead, which means a more involved code rewrite -- and all because the two cycles were added to the display loop at the beginning instead of the end.

IFLIs bring on problems of their own. Sometimes the display loop is unrolled, and looks like

```
        LDA #xx
        STA $D011
        LDA #xx
        STA $D018
        few cycles delay
        LDA #xx
        STA $D011
        ...
```

Sometimes the loop is only unrolled over eight iterations (to cover a full character row); sometimes, however, it is unrolled over the entire picture, taking up thousands of bytes. Sometimes the background is changed within the loop (LDA #xx STA $D021) too. Fortunately, these loops are usually written using a code generator, as part of the initialization routine. If you can locate the generator, you only have to add two cycles in one place.

There are also different ways of generating the interrupt. While VIC is often used, the CIAs are also used -- if you set the CIA timer to $42C6 (NTSC -- PAL is $4CC7) it will count exactly one frame. An IRQ typically takes a variable number of cycles, since the CPU has to finish executing the current instruction before the interrupt sequence takes place. By reading the low byte of the timer in the interrupt routine, the code can deduce exactly how many cycles took place since the interrupt, and hence exactly where it is on the raster line. Fixing this type of routine involves setting the timers correctly, as well as fixing up the display loop and initial delay.

Conclusion

Nevertheless, with a little bit of thoughtfulness and persistance just about any routine can be fixed up, and as usual becomes easier with practice. As an application of the techniques outlined in this article, Robin and Steve fixed up the graphics entries from the 1999 Mekka Symposium; you can find them in the Fridge. And if we can do it, then it must be pretty easy! So good luck in your fixing endeavors, and see you next time!

::::::::


```
begin 644 flifix.zip
M4$L#!!0````(`(%>VB9V^2?#&`$``)$!```3`!``1DQ)+79I97=E<BY.5%-#
M+G!R9U58#`#/]W4W01-U-YP/9`!MT"]/PU`0`/#K9AXA"`8C*QA()C!;.D+(
```

```
M@/)'5NP#5`Z'1-817$4O2%)#&MR)?0`<<B.!G!PP)A%/%%!D(R2\*^0%P3WU
M>_=R=^\<-:L`+C>\K4TH(J)3G!N1@R[C,M,0ESB#9)63-;X&/X5AGS_RALH]
MU?/48,3://$4^=A0=(Z+95)8+=,*SC,UL<)ASQD-7GD244N26I(7",_4-X6I
MBC4.Z4P:@NG6.9FAYEV%WR`?3^_'TU_4]?_G)^KZRJ62E%2F&NV:SB\W1UI3
M3>A;K@OW+-O"?<M`>&`9"MN6Q\)MRTBX4_"68PJRDKE(S`2!?'MBAN9/\VS!
M*#N,^:M81^YRIZ6[&9!CMIC"0PJ/*3S%_-Z%/_$-4$L#!!0`````(`()>VB:Y
M!>XH=0$```$"```2`!``1DQ)+79I97=E<BY004PN<')G55@,`,_W=3=$$W4W
MG`]D`%71/4O#0!@'\">-U21%:,5JJ@@1'!1\[2#5ONGFT$%PZ=A%L)_`;L6A
M@^*%Y`-42L3A&8JN#@5'Z2#/6+6"2T4ZB"XN::EW:00]CC_'#^Z.^Y^DA)0=
MN(BO;VX`0`G+?++Q-JHL*N,-FY016(280<:^@DF^S"-0KV1<]HUHWZAU#$O-
M%^#/<*`!J2JT&I"NPF,#,E5X:D"V"L]-ZCB2&243R)PA,ZXTS6FJ\[M6V01A
M8CE".,LDP@"#%VRQ*4*HN*A5>@[4X!94V[V_L^C[3-,TO2C50K;+<\[+-2]W
MO3SP\M#+8R]/;??*U8M6Q;4";[WWUQA=UP`+MGL^*!<PP1_%0Z>ZE&3SM-#%
M+2%[0@*^;`O)"Y%]$57@D9`17U)"2D*"OJ2%*$)&?<D(T86,^9(5LBA$\45R
M@-?'N^/%>:U]_>NWR\?'0'X8R([2#/=_URM-ZO-NA^>)!@W0NOPP%B46&W+N
M4^7[ZU)[*4Q!_JUQ)7?2A1]02P,$%`````@`Y#;C)NBB^O*O&@``A3$```P`
M$`!D96)R:7,N<&%N:6-56`P`00A^-RL(?C><#V0`W9IM;%O7><</:9*Z1Z1Y
MW71MDFV%619.)`MI]M:N:^LZVXI.GH4N'_:A]0JCP#XL@H!A^99A"$P+\B9Z
M<2V[HBB2HJX@A)0N->D.ADCE2J+X84)@U0.C(FM=+]6<%29KRA0YV]MM>.V1
M^S_GDGIQ)!?#OLW'EZ^'YSSG.<_O^3_GVC;)+;W;-OY;O_'%WV2,O:&R2[[U
M2Y]=O_2W=M_7"Y,O9A>E.'.H<IS]Y`='''DRPT0B[=7W=G/RMB3<&;/JWI$C#
M,;C^"WW@2Q'6P(O\-Z3\*>D'ZY4>M_/[;UT8^+L+@W,3P>#=Z[YO.M>,]6?H
MB:VWX6D:5PK7I;MME1>=ZX<J'<X?K#\87*_^\ZO.]?>67W0.O[3<X1S^O/J[
M%VS?ZF%'9L^^5_D3ZO>J\UU;Q;=>M'W71W\.R8>/R$?XH4//^^S/^QQ.^S/\
MDR\?L7GYH<_ZFG\,QI]CK85-LBS[*E:392?B[%:6?2W._C7+3L;9!W?7"V],
MVK[WJ^N3O_:]3ZQ/GOO>X9^J?WCI5PZI1RY]ZM"W>EXYHG9=>F9=??<2VU!_
M<NG3Z[X_.^)]?KWG_.:$=W#]OVG<3UQZ;CW;]LRE3ZS?_?!?UJO?-=YBTG,=
M7^K^UFMO_']]-GSL=[[PQ=_]TN]]^2O_7Y\-UNG+'C%D]HJ184>.&-]C@0AC
M&U<9LQE.YG*Q*P#GK/%)UKA]N]%@N=L;S&4X',QP-!XUC'9VNV'@C:NQP1J-
M*G,Y'#9\6<>W^,ZSP9C_]@;&V4`GE]/IM''C4^P5QAP8M)%C/V5,<G+ZM`WC
M&Q*K-AA&;#1NX^LJ1G2YG)+3V8;/,0`^KQLNUH.KX6&,^R6."9V2)#'V(\:<
M>+#A=YA.V(RQ\2-)XBX'ITYXXSC$6&YC8Z.!1O,P]JA!9C>H&W-PB4N,/F6<
MT7(:Y_`M7M29>.C&%Q*G?EQF\`,MP\Y8X#SYZPK\Y6*?H:E=HI?321/BXC38
M!F;;8)B:7C?P!I[B+FZ3:>@&N\WL=AMGAA-?8N4P`2^9Z..U>>$=R6Z#EUQL
M(RCVAUW=H+4R=.`VFXUY[62TQ)`9FKYM&![Q>VYC]#5LD.S<QETN>%FLD':7
MHP/STD@V#-`.$[@#/9S,C@L_H>61^[`D,1&&X!C"X7#!)(?E)WIVH>5<8JDV
MF^&0N`/CT*;>;HB)G"ZRJ64.QX9QAKE=3JP(04)=N`,/Z&#'G%R2[<+1U$-B
MA\37-)T80?2FWSAX&\.&PEMM^*C-M=W#1@NR>5V(2X?+9OPJ"UQ"@%-0?!-N
MP).O&8+57..GAA=C?889[3PX:CS+FPL/W*)-#0[9:`/Q7OS8H&D=XFK0'NH*
M]I3WN=TQ;N%`ZZ(E,T$$OJYRM[N<SW.+"/J:EMMD0A^"`T??>MQV:90;AZVA
MM]G0QQ@?&;TNMQTV7)S@<.''57*GPZ(#H\/B-?D704[QPKG`@ZSML?:%&+'Y
M=;P)2GQMC`,/.,]B1&P=UL1;#N'F93TS('$SBI_R-2?G()TXN<V)$][B1#6U
MU7*9RS4N.#E$TUJ<<$*$-SD)FI>]^7+9C(X2)WPMA-T,_)!<^B-NL^*&4($]
M^OSITYPWXXAU_LS;@H7?VKB%\6X)1V?&W'JQZAVW8.$JE[U<P,(;PB>`17>7
M5[@LU\:MU:O1JW*5=G,CP@,;\+P@9BASH[SBK=:JXX:3,[6S=M_+:=Z_R36L
MG<7NNZF+<K0ZKG#3E+7H`[G.*98Y+1APC'`7IBK)]ZOC=44U9<YK=>8VVG@O
M#6$XKKFO=5T[?6.>E\;'O76YL]WLZ*^.QY;[]*XNM[NK2TQH.`:D@8'AWIBN
MZ)G5?/VH<M7D]VM][AYWS'!UBS`RVO(Z'(0.GG)=KLL?R9K\7CGOZ5LSG.4U
MS_$-]#B3X4P?ZD4/PUF/!F4E3QW4]D0BP=DP8S7#]3"#07@06!JN2?NITS?*
M^>6^0R=/WE[SP'>WWGSSV;8WXZ?'N!X<A7>\U>K1H[(L<Q?&S+O[_KKMV36/
M\2P+;.Q"*("\_@H+T$(>60@%N?'K<!]G@.C3V-<F-@'DY1\11K3GB..FFNQP
MA)W3E4=@(G/#$V,'4P2C/;LHPMK1ITE1@`6#U^5S-HLBDA=N&6=_E-.5S_&1
MX/7'YWZ(&*,X>Y*DOVB2)/''YT)/DM3=).DX/>L2@XOX&@;9H*$VI#TDD=ZL
MPB_<E-WN`4FM<?[GZ/8=D/3!QBV$Z@=/D"1?[BGGN:P(DO@7V,=).D_S:Y>]
MY?SJ1Q>Y4!P.CK$71-)&BZ0-01+33\_'VH@D6CKO?.!E?I&E-G@#LD.+PAMX
MTJU[Z_]>XBV2!`M/DI3F-2`NQ(L32.,MD!B!)!%(Z.2MUZHE`9*=`[:F\'"?
MD!>7&*<.D$I'^449(-FK57T;I+X1P]'+,YERNG0474"2IEV&,7G$OY/WCM#D
M'==`3%>77?<L\U*IQ.I*IWK1?O]N*;;6UVFXKHD9>P:Z'5)W-Q>DN%?+]:-R
M)Y@TW[E!0[D&1"P5\F7*1SJ\I'`WX525-1-&-7'*KXU]?H-I9Q+M("<SS`FI
M&^52M5J]K$UKO$S(U+0S9\RUL0%`=?+D(0!$Z/'1T5')6X?)U5/SGG+9W??^
MPX>'U\:(K)O/OO[Z3>X&67RT#W/685J]JHSW]GGR90:R7O\X6>RI9#W+ML$2
M(1`<I1#HS1P$%LF3V^-Q[Y*GQA-@P6;/Z`Y8G]\+EB5/00NLK^^`!7U2_)8^
M?09EB^/C7!UJ<;7V^)SX6"2%QAZ!^CP]<\)*XE$51&&@?:AB@JK7B"KH$TD^
M)WUZU*2JL9>JZYZR=UN?_FD??;(15::4+WL&Y-$F5?P`JF),GW?SMC#)5U.?
MN%U4PAO\`Z%/'UCZY`%5\I5=^E1_0I^X?H/T29''Q?:H?+<\[5!%VB,WY2EJ
MASPQ*X\UN+U9M(F!O`K41Q;Z-/I`]NY@E>ESN5PCJ$EV*92)>M7[C644*2XA
M4?T=]OYWKEWKT*%1G(_GH5'JM%V#1GGZ.NV&H\LNP%KM<?2\->`8MB2(-$KN
MK)F\YO;T+,=6'=W=W0[!UMF"MU#(>P5;EE1598FKLIS/>]PEK[>,_:"X,MM-
M33MC"L'B&8]GW.NMUTUF<H`)R3+;V\TUSZN@Z]#[)TX\?#]#W`0YSX,N5_64
M4("^+[[__F%+M@SGLU=.BRXD6K)LB=9(1JC;\VTHY)[;(>OL'K+(G3^U(Z1W
MT.('HA4[&"TL=@]:>S6K-W;#L[H6H][;9/F?D*P?-B7KU;UD?:Z7R#IGHZC<
M1[)>%&CI0K)&#RK^2++\?%I(%B*`)&OL2;C^#6Y!#?H"BC^WNT?BD"Q_4[+V
M*_YX3;N>SY>W)<MY0/'':XN>O.<_$79/A:L7<+DSI;]NLR2+,Y*L%ERWA63=
M%G"Y/<MZO7YE?%NRY'TEJU139)*=O97?'L&JDV`IR#)FU$XEI.7P)EMB>X5F
MR4*SIDFSGB'-HE$MT<H8#I>.RBA=DF6"*VAJ"D3K73?BP#GB$J+5W]\OV?OM
MNGN>Z[&U,M%UT7[T,H=HQ3KLUZ[9.RS=6EU=]3A6N25)I%M5Y:IL:J1;;K`W
MX.BQI,MPGO4VA>N&@&L:ZG89S+ACWH+76["DRW"`K\PPV`(S$M32Z\4Z>WK`
MH&HXM+6Q/Q6B_+XE6L"T:K35.9P"7'Y\XH1%%7O]==OKKU\1DL4MR;I_OR[+
MXU1@`C_>9MLK64^`!5]N"+!&?PE8/-/GYD_3K/SR4\#B_!0?W0:+:D'[+K!(
ML@XW)6O]"<GJC8U>?]QFLRQZDJO/;4O6]?=&+<EB+<GJWB59=GZ1NX/666F#
MK7U /\TZ(C0KY[XQ$.2@S] 4K/W/5/(/4&!0- J4'W2FXK5WD'D'+O\2S2*L
```

M^MSCXSN58'5'L_9BM;);LZ*UCVN6NZ59(A,)T:KO)8OM%BT%16_-!'ZR)5LM
MM'`V$J)5(]&Z8IJ:R9^I>N>MD[P`RW#UN?OZ[F&R>MW2+`E\OKOLYE"S7E?&
M.EEU2!T=$D[47#_M(=6Z.FK'B*?7W$'#U=\\5O5X>GKT79(%5F3M,ASG[C&<
MJ]:YRE7PBBX\(T(<T&GRY7P>]2*8.;OF(;$R7!IR/Y0*]1T.<N->7&NKJU"J
MF&::4"K,=@O<G#AQ0BA5\^PD&XYJ+]07_7X,K1+1]?RS-IOM=4NM]`PQW%0K
MJC%!E9O;UG:KU3?_]U!Q"ZK,4Z#2R2@4Q`<A-<R#>Y#:T2K>U"KD$Z.]=9MB
M%U,CH]]Y?.X\9T\5JU'^UF-.8L7WKP/]'#N+7C)!Q4_MJU4X=KZ@FI=/W_"@
MOH@*J+Y]D%:9VEUO/J_*W-*J[^QSO!)0J8L]J^YI>5NK2GQ?J!!S[F*IU(2*
M`ZHJH+)N5&P(J#8LJ,K0*OD*9P=J%=T1@GS49*J:V:Y",+B'*7%VDJ%6JJEH
M6O1J[0FF>@W7CEC)O-T2*^]I;NF8H.I:G^'(N/M(KFI5.F/!*_SYJA=RI??U
M];DR?=1WH+^C7^HGP1)@D6!=Y<\<_8B?AF`9+DNM\CVK'L_J7K7B*/+^V$-J
M93A[UL804F<*9YW.L^.66O4)M:I>EDT9:@6X6*%0<(K[%B!(EDU3'+2"DB3!
M[24O>;=G%7K%'VK:R;4Q`NS]PS=OGGB?;F`$Z0B%@O%H%<3V9J`0'O>/#Q]^
M4^B6T6;3RZ19=+2KR[;[LM`LB)O'[1ZQK8WMKUFM6Z7BF/44NG2Z?]&;P?GZ
M(+JP%9Y\?O@`NA#62"-[#UG^W8<L'*,N!??%*RXD2_[%WS5W=5^ZZ.[%]3'Z
MF(L3WCZEX%7U<N_IH*3N:!;;#R\?,S4<8X,(7)2"=+#+W1)TW7J"+OENWENN
M:4*R&/_"_I*%`%E<[7%/7R:Z8%+G`7"=1EJ\AP1KW04$7$=Q8M@'+E(C4JQF
M(1A]7I#S,<4J'57N6XK%GZ98H$O<XC.UCRM6+R1+J%I3L>1V*-8WYPV+JZYK
MI%;N:QG8P^OCEF!U@-#,LELW++7JEJ3N@8Z.UWBL)5?*5?[@OJ9#KDYU4Y'8
M+<!"B.;S/9[FZ<F2+),*/$_9,_^N!U6B9\V#>$J<<9XY4VBWT&KJ5AW40[?*
MRVXU43B;$,(%*7OX4#MI58(7+UZ,/J`;7YW**<<J,6@X3F(/0-;A-]]_#A
M/=)5/5JM]F8RI%U]^-HBRV6[HM_8)5SW6\)5%FBQ_<I!7_,.QO\"K=,?1XNV
M''`A+V/CO>4#I(L'U_[KW&CP(+B@7<''.]K%AM@3=(U(UQ^?DY]6$))V7?=8
M'^]#%XG7U7:3Z.(U0=>IK^Y7$=(]4U7N=GCRHS7%J@B_S7CK7Z.>P,O,0[S:
MFP>M4Y[]\.H1%:%CWBUKVWB-[X_7,+2K7B\U\5)1-.V+%]WVJ\NEEG817OMJ
MUW_5T$G<3D66J#Y-O(YRNL<&O)2_W'/6ZNW%?O0*^:J3?"F3FJ;Q^]7\?,R0
M+.FB&W]]75T9#Q#SCG-HU[2&DY;W78\;0@7Z+.TR'-U2]VLXQS2E2[DZ>O3H
M1P19;&!`ZI`&+/4R'&5/>8]XR0#E(QQ^YHN$7UXP=L9P)J+N;?&BNW^\)FMY
M@.-6SYQ)G!%'K8?RR9,GY9-"O-3IZ>G12?/!@P>341`&N0%A#]?&B+";-]MN
MWKR9&6.6=C5O_U5[74*[,OA>$/8\$-,]+?&2Y?L@[/[XB%"XY7DA7C*C^SM/
M$;#'/N,UUGY,/19)1!(L,!$P'$/&A^SXA.$*I/#3@ZY)^I=%7)%D5$UV)CNO
M#`U-3.#GTE-^\W^Z['@PV@*&CQE2P.AF+W2^T'GL>.<Q0WH%.<)NYYVLO3.4
MC"1"J50RLK-O'1UM)<*(+-#GPXKQIOB%^V?M]YS&KT^T\S.W.__/+LR]ET
M-AJ>"VM+6K::?50Q2XNE>G5I+A914R6]\BA;71@/D).&AH:,OSIPN"/,W@Y[
M^`O'30R1+B2C&"R[E(JDDQAEJ9)*)`U'))12PG,Y6!@.W?WP3N%.L5A\RIB'
M8:(]55K4=76Q%(HFHQFT6`0M$5'BFW$S;E;2,XK5E-RLHBE3RH@:08L.&\Y`
M:G(0L*7\]-@N7@<Z+AJO'CB?1&LPG#P>MEHVFMU"JT8*:"6]6$E7S'A:3!F?
M"<5"FH):4YG*30Y.#DX,^OK]@\P'Q_M9`%/Y[/3HE^P^\:_>$QV#_G9_.S)F
MP/C]IUC0Y7[Y);399"F94!&>V6JNLE);J<TNS=Z;O3<W4TEO+:25^%0D4DSJ
ML:V5XO#F<&)Z<MJ'/<+LSS$_IO"S0;O4+V'"0<.!+XP_V#7C'S%;YPM=72_G
MTO?^XQ_^86SL'\?&8#2"Z1B%@EFIJ81%)%%*TKQHU=32W-(<9E]0M_2B64PG
M-F,+6WIB:X)6'?"#W@!8LPO.L$9_L#/8Z1],#1?F[\QW3$P/'H<E6.B.":^0
MH]U=_.1+>C%2&"F$4F-:6IF-*VD$X&(J$DG%9[2EK87*8G8AJV0U);:BY>;2
MJ4ATL63>6]R\ERRFT182H?3=]-W)83*#"3,^P22[^(\,LUT7_>W]DGUYS&K8
M"&R%G04"".4)WP1LV+'G*Q1H=FGJI70V7:C!G)2:2LXH\7A\=CR=+L(>^ILJ
M)2/14#*EAI.I4"(Q%5Y<G)F9RZ8W$=Y+\=343#0>3?]L^N+P1`#F#,(:,"KB
MP2[Y+W;>O7.G7!X;^D=,!3OLDF^"6F"O)5^$)9V2?\('.Q-%K.^E%_F4HL$N
MS+-9V-PL%C>3R6@(L$?4T%(U6W]4"<>%V778#683L6*V],Y0?[]T9=H_.-'O
MHZ$I0\W^O0A&-CP$^R8[@_V(CE2Q5`Q0KI@8WF/&;PB:@?.Q#S^\,X^,D*2%
M!`)#I805K,5B8;.(%@G-A#5L4E1-186/'E468%ET>EK7-.R7?]HW*-P!;W18
M7J8H\0\>3R)(@L>0N0Q'NE@K%LI+&M+"7!@]Q2897]IC#Y!X<0$3+6@+6JZ:
MK=Y)5Q8W]:+ZM2_7A7/B"PO*PL+*"A`M40J+A);B"%NMLK"5+2S$TY'1!#BY
M,DW[@I++Z,0.V`?M"->19"PV%0M/&8YZS:S5*NGZ(VISX2A:(AP8FA!MCSW'
ML13#82>#D.XJ*S`H6\E6BWI!I99-6VUE95%/J4GU4=8DL`K8KZU,+5,+A[49
M#5GDRC2U4)+^XXUP$L6%7Q([P@91W/GHC.$;%&'4+W6V=P9?5(^-(.THAB,W
M-:6$%>0]A(!)SDC`6K(9UB+&`X$]%G?@T9:PR(Z0K719UN[8:PKGF:58?`GI
M-#M;^'DH')^96VKU"R5&1/,/;J9+>DGW3_M3M5I]Q3?HGXXF(TDIX!^<#$60
M<?T7583#;[]WLI:$/,V.S";4A02!A>U69W/97!UPAV9H\&I6B,]R-1L0$+,=
MPP,3@Y,T,!7)-`6\,(B@_=@330O7B6E-L11M=GN-M"=Z0M]>8\EJZ60TA0@)
M)\!T-*2&6FNT5C@2+>F;^N9BQ5Q9R>7@[W!II%3"S[*AE&BP)15-I:)P9S@:
MF@F%IVA7<KE9;78685'$CY';,+B0UJVLM@!%B86F`A]^>'?S(S--Z\2:2)3H
M$3F;'GS;KRAS,SL69'GB7J%8+.&*)$-1<452R$DB+XG<9#VKE!A2N)*5)=+R
M16AG>B4^JR@KRE(\!M4HU+9,:BUO)"))]6*POQ]YM6)FJTN1YF"%".4Z-3D2
M)5^,I)J!@RTLI$MI(%4JU$!+':ZC(4(I^",2C=07Z@!0W\SE4+_5:L59J"4\
MD2$OS,TL:8WTUM;6K#*+N1ZA5>O9K1SV?B9569K++<0+D]9B4JUG[#E6&XGX
M6XM-BBNEBG>A6<LYI1WGM.R'"U*B(**+BACLM'!.1`P0Q0!158$PY)87%FK`
M&)K[]SILRE7U.#FF;G[MRSAU!OH!4K)4%ZF'?(LH5^$9K+)BUAOCRGA8"4?(
M!46U6-6U'+614*@`&=$7E82"9*ABME`X-Q-!!I^9F9VM9;>VQN.Q6*R4UN.S
ML9^/A&.*%L]EI[;T$@(0>"I+"A4@&%+?K&V:&"R-EA7FJ^1I6E](O+$V7JQG
M9^W6)_LX9[>#FD^6<T+6H#N?Q^?B*QFTSDD?Q-4W./_1A(!S5[5"7IN%W^(S
M5FO%2"QL-7()-;B,6G1K)A5#BZHCA;29K6=CF^EX:BFRF:K$4H54!;,N15+I
M>_%X,0PY":6*Z7HZGHTK5$@BP95*!6K%`H('X9/;0FV6J\)3\`+JR'2Z:7K+
M/:KE%O7)%:,PO'#A`KA[NZ,#]=(O\X_XC`:U8B82HHSU:"N3S8@P\05$\3=)
MOO'M]0_5<2^[N]QQJUJ,51X!#[0IR,T4_K;\,[<XEXH3M:AP:1?K2Y7%"G2C
MD*W^O)A1%H1S4HMS43A'AX>01+%=J=FP1@,I!"J6F2@5+?<04.%0+)$L+9H5
MS`=04<1J0!N4->(+S2B*1"U,FBLF5*Q%7[@0",)!P8ZWX:%`Q]O"1X8#HX,I
MRYO4>;>+++\[K6=(:8RN1&0J1&UP:'+XRO"5MX5[D.%W>8CS=GZ\$X7,%((9
M/X:UXCDVE:LLD964[%+Q>&LR5<RP&6_-GLXHJ:5*([*@%L3[1*R!FF-*3R%C
M#4_>*63-27NP/<@C",=8>$XK)4M1*T9'*+-%<Y1]D)G@.Y,2(&&;1DY"KH!C
MK$!2FUS1UEOI)(K"_D+@`KSS-OD(_HDOQ)&]TJ%M;U"OG3^2:U)I/)76$[$\
M-!.S #,5NFO>H5:3R#TH`UH>0AW0'[P8Q/%.-1Q3&$7H.YY%'*5RQ1`M1:LO

M41%2JBFT*X@%$:PA*UU2LL3<25780#HGGA%JRS\/(R9#4\$7U..)1"@13Z^D
M,YG2)J7EG!:>HZ2&.LJ$*X2WB#CR%KC"Y[7*)HB;G<6W(@<MQ04?&!NRL[2)
MY1N."X'SY_#1QU4S\%+"_&%D_&O'`]9/5.42TE%X2W#14;2:^&@5$JO(Q>B
M"*]MUBHU<`;90HN;_7"/#\>.EH>"_3BR3X5C&K$FBA_LYH*%63U"H$%LJ!Q,
MBMH=F$'R8\VZ0TA<*J1&J72%BV;H<PAGJ@A0`&MN5FB3L@(X\1V)/L1.C5!9
M&$FH22%BU:PV`ILQP:+X')YZU$@CH\X@'$M%.D"'4)LC4<5IU)PRI6+UAC-U
M_GR`7^`LA*)GFD+).(>JE-G/L?\!4$L#!!0````(`/*"XR8U.FMEEG,``/IS
M```/`!``=6YD97)W871E<BUN='-C55@,`&>-?C=GC7XWG`]D`"VW94"47=<V
M?$XP=`M2TMT=TB%=(IV#TBTES=#H!0+2#4JW=#.$2(,@78/2+=V\W,_W_9A:
M^YR]UHYU!`@#&^,(R.'G%>8#`,"'DW>B5#0"]%6GXQ0C&P`/$5Q^_=XAAY%]
M`0Q-7.9Y#'5<^V7?/7R:N!CCQ1@3P(B>.%`G`2%`L0#FYLT$>/,67G#U%4S+
M`XZX2OL*>_JDO0;R@+3_"_VXFMBG)01'<*E.R-`R@57CE?_O?6S@_X*#B'CL
MKR!:YJ>O3[]+GEYE'?Q@%&T\[--7R%.XB&O[::ZAB9OHB<OCBJ?AP<=CQAW@
M_TO&3<L`MKXIXHK@*KS=X([@WO[.]?3PP%/)QU.E(A&@IU45WOQO,(+;4!W`
M5*Y4CCT$@R<@AQ`P>\2E'7S[$^QR`D(+@..)ME$8$R?'V%_1M___$/O0Q#D9
M'-!#P=?_)H-D'.+OD/K#I5`K1RG-FL[-_0,,:YP]\/^2T.JP<&GVV35&*$/M
M8>'V>*$Y'`KU_+HA^:<*`)\?\&(R^&5N(`7UQ_4<-\R]FEQMMO/:7&V5_I<
M;?OSQBS[$"X*4&OUIWBR7Q6MP;&X,QLJ3B[FNAW]T5MSZG8E+]0"[/(H>FQ:
M0P/BV*CQV;\<XAX8X&QS:N4NJWR1PS0/LON*.S1QS[@+#VD'\&/(%`N:ME$#
M&NCT37M"?)Y2NWIK9\^$F`)(SIL]$^-\+TT^J]":XM9K4SVG(E8)#<E&@Y/I
MRY:[9:"'1#\[?IY,8`%8(+R@?U71@*,/VUIN);J.$;PJ!;1(G6@#A-ZZ&$RN
M+88XOIV]X$584C`B<%U,@SCY]?NA9"OWI.-G2U'L`>YU2AQ<"4.V2%MWJLK7
MPXS/-A#N8EH^^.;R*21BML0IS@,;VM)@L0'W"72N9O=%<JHR"2P6NF%OJE;9
MX:]6[I7'7%E1W#7N%U$\E@E#P\@A=ZI.NB%61#W2K0V2W!72N2XF#[Z-_IOT
M:6##%^7;@J?CLR'U^<TB0XI,`($MW5`X53G=4(V5>_^Q8E84^[9[*SI'>L(P
M3S>W.U4U:)B:ZGN_$A<^&N:`^V`4E[;[[W2JA@`26[IA4ZKF_.%H*_>%8^.G
M],KNQ^C<UPG#P=U![E3M(<,75%VOJ+XF))-JP;00UDCWQX@:VX#P&@)0M+$F
MLZ2N82KA7U;JTL>)N+C<ZJ[QU:/;>\M_][]2)ENV1ZY7OD?#S57OKGM:4C:
M]%9:Z@"I_YANU1,*'NFT_C$^FD]0&Y>/FV8Q*3.S$=!QKC$SLA&D?BTM6N.B
M+?SMPU;`\#/T*U&07L+WE/J<K$\R.3$RT:$%J=1##M:T)[L=KA2Q6`0=4>-Y
M2S<?QO/:;^+&_9O@L?S1FY2Q_/N;G"?QTXI1R@'V/$'(O0T$0Z>;GC;`:\!
MEJSM@?#MVJ";_S3/BH\H@@Z+MS$Z5'"V0QT_:GXL7CT[`AJ4<;9IVP^_*2%%
M6]^!MJUTK10!C)+0U`/ZU'\,HI+A>=@Z@]<8OT#FA:L(BD3:'ZZ!QE*L"HG%
M2`Y&=%G)8*AN^@V3WS/N[KTE%K\DMZX/I4K85R@#<X:#`[)&U1)6,29^AB^]
M>83P5[_]^.RT60>]R%C>9=`)4>']X-O&')MB^>3Y^[]@9LFXI8.WGH4NJJ6"
M4+N7+KV5.4AD=#E#+5.).);%>IM/)FM/-RG*RP=[O;1+AN(#US>9@D"5YR4P
M)&M19&[?3\U&2!3HY3\);N=70S)38:]X3R,20Z1<WVPQ862C*>(>VL`&XYW5
M?NNP/:1#?UO>KB@_PF*$L1U@X5`?'\2'M^@LC]VT$[A1L=U>'5AVJ[AVJZ0:
M9_<:_R#ARQ-8=EV:T./C[<$'2Q3(\@T\=_!^`A^^.>`F-">:E.;.B)3OW_[P
MVS0\=JQ>:2F=I5?!&H`D'G\1J<=%8R/K]NFV;_VE''H`"?V&I<RFNGA&=6NG
MM2_KGHO&Q')WA&P8+.G;1D4,);_9(Y%\MIMKA.F;PHDBY+UC^K8A?5FJOAY-
M*Q+5\[Y!AT'PP*=.%?T63V,7-X;M_)DA,:B/4+GR):-.(J>^]VL4&Q7LE^]<
M(P:A-T]_ZASF8M*53DF;W,B4/,Y^J\]0,4TB1"?04I#2@>E'1@!(<U&4`DP-
M<F+*S(O57OAC7\:'<NF.;CO6==XNAT%YENVN0C^$WG_>^T^QN-ZNLW"\I-TY
MOG*L+%#I4V&,<UE@2'$H922O$_7!2\H8\(<2U`1C3")M:S(==[2[HAWI"L/L
M2=!7U)2JE#QZ+T0U4:)ODN,Z))\!8>L_9["$>X&V^%NV70L7]#HB:(L)(0+*
M>.M3C#X35_1)LN_]S0MW=,+9]A<?\I3ROOF`@_O`I%_G4+F8+1'D*(BZ[K,_
M6["92^H#;H>QS1]C3KC'[FE@:U$I@JY[7**W:=6*^CYKSQ,V_-6YBB$F"TFC
M9!U:T/$*.V^[+;NIAP0WD<1EJJFOLG]>BF1]C@OO5ZZ.@LQ>\Y*DHYSP,)1?
M>+I:;/EE'^9,TDF)M!%/J^!PE9[ZF^?T&)2V)QJ6CN6*MV`:HK=GQY`YJ#R^
M_>OS2>N[$NB7B&F4PC_/%\3+142FQE;8.A?V4HBIG9>;,F]V1(MD7CQG>97A
M:7B),(KXQHK$5<G^0S["4N`)9/_:QBC\5Y$&2[&ZLJ36L7(KQSB43YCHX=.Q
MX2W*W23`^.6LF@A,6&+\BGVEJ_K\1#IH#NWC&-'LTGJRO%<IBLJX8HO<<=GC
M8_Y+W1V$L]J;@%>?_,=8(K9U/F-VY'?21]S;$H<3YRM!ER:6"2UF>-*[7SZ#
M:VC:N6#P.(I1$=.I\=+_"46%GM29?L:,#P/C89#?'7\O^(R)"`.#-TY0'?ZS
M<8C9`-K2!YQ<%&K/KT=.+C2T%R(O']8'50C_@?:J'Q;Y4VD`XR/FD.IPC`;N
M6%P2^-.7K(*JYCX"[NKX]^X>20V7HAF4'18T7_L!F)@SJ1VNESHF61[M'`$(
M_)6CV7;\'C4`Y,LV8P,$`(@#M,2,7WH8TO1C7?`*&YD1=']K&LQRLTTEYM;2
M_6U'`U&^;^XC`<,*KHZJA/M0/-/3`P,NAU%1FRQE)Q;X1;K&WZ_J1=@7?B=(
M]+M43O&N&VFH=0=(8?69DYSO44B&)J`3!LK$O=]?#RAO^(/WTN9+YYYA106G
MXZ=/45;R\D;[G`2+69QM/T<6CXZ."5'M<68W()-_^`_RBYK8W\XO>.>KY_*.
M'RT$P8]4I[[XKB+11R$ZF)@8)=%Q=UC7=_3=&Z![</N=_$.7]`!/@0?R4_%5
M:'`QLCFM^KCK.,.XQHM=4?IO>^3<4A>OI,<6<<X$U?%Q7CS"#:XM\Y(L/O$X
MC]!1/LVEU,450U146]O.B>OG>H!2@8^2)8F2*"N7J:"SDGJ04I^?4KCWBZTO
M/^]A_)6O3JX@.,'&I!7S?^H8GC$R2YJ<_BA2S/+K%__M943]?9I1/TX;;A-G
MA--MRS`MLGG:3PH**2[Q"110*K-`B_C6\=GGIB].DY"0\B?#-/]4^0O;*R`T
MC30X#(@/)@"QK,'10`#^E4Z@UM')^=7M0\B"V^G%]=UCZ/J_L\N;>T18M04N
M;H<2<FX)T(I'#@5@(S$1(?Q$@]S4#/>/@`;P4D)*1AY@Q<)!K*E04#%VSVBM
MJP*B"HHJ`'M[`"DG)VUWV>O'NSO+N*:'R_WB`<TYRW6[8^#HI26K/==3C_<
M`?_A;[-FR1GNC9^12UJ<JQWBN7UYNT@U?G]\]2%`BGKY*I@R^D-N]/>C;_?A
MH"Q,:NLL="EBF9=227?XA5KQ9R<OB23PB]RD&=8$3G$5QN-_*X[GJ=^H'DL^
M"\`OY/G?.&$'5H%6_(#:L522!'[!8_SAB6C#@P%HW:9<T9>&^F?HASD?YL"H
MD-5\E!X$%0#V#"+Q):!CTK8(]W:*\+?Y;J/@Z[6'+^YB0.^1&M%U[&><0[1
M/F+L3UTS<O1Y?7GYPO[X2LGRF4"?-Z'X"P1^N'?P9_E6O%]52KW?4<87U#!\
M2Y3YOH^"Q"O#O_GZN4_CC.VJI0,__RYQFDMC/_9_&VDWEOS1_^Y80?_34%@%
M]MW;L`K-L*ION7?F82-RW_2RV.@'I[@:Q2';$C\?QYE245X$-V_>9M&$08CF
M/;J/":M/E1\9[";E)R8M/A^+(6*M?*09ZXZW)J[$45=BJ"L)U)7XV4M\:8)M
MW<\G4/5F"9QF"3Q)_,^QD+CX"=%K[+"N^.W9J_T<>C<T_)BF&)'_FOX3^=3T
M2>1CTT>1Z*9HD:BF*)'(IDB1B*8(D;L>'KGW)@2PKD'ZM&WRX711:5&)?B8?
MORNTGF;10:$@#39O-[_(3SEJS&_%G/']^S[\E?N\9!ZVQ//0*Y'TX_Z-A%P-
M^^LJ^>R7ZZ0Z+J&%L'6)+ZZ\"[4B!*Q!VMO2^';2>"72>,[X:N^^AFECA;%B

M(0*P0H\QI2:.5>O%I:>L><>P_37.G!:BK.O.?6%?8P<<F.;`I1_25IF68OOT
MC&@P+],<6)8BB^\83;!*+STX/"5(AU5?A)Q]=!=+Y?OEA-(-I)ZC43*@JXAR
MIU(107@RF_U\E7?N(5YF<!K.^'8X@82FENZ5,A=^R)[\WSOWB57*>O$?!_@A
MCN^")CJ)R/*^T[1^YF/'1]"IO3.?8B9I%.^MQ`\11ZRFS*:A^<-7R,%.XX\W
M=H\O,P3&N_]#6S>5;U<5(6MA>AW=+X1/:$-,EFB+CSK"&2,\BUY:/S81%8?6
M!TD?F/^T"X?-LJ"3;>Q1CA(3PUN1(D4]5T['WF[N^((R.*1A(7@[.Y&K#"'H
M*"?KT0U##(B/2.;+7E/'U?ELW\N0%6%M,G2PMD;.REG5EF90<LMO*V4.Q"W/
M/B-Q'8,>@GSYT;U+2AW@_5LR=_!B0O(@TY3H4;.E)?XW3%Z++CR0K81%^]6%
M-4KEM#JS0_IUZ5>D!D<&F6OM$;<S([?'/U(D./+/80NG=.3IZOU?%KI:#]PO
ML2'<'RL+U)4+OK>/(8$[M?J0!NWE4&_TK3F_(`8&$9'8[0,J-[O875H/;*!^
M\&ZL'M-DI`6M&?N__?W&QD--P8NUDO<?/@/;R.][E-[D5U0]CRY=,SYH?PXB
M>HW=(69?8>GXSUT90N`9;-_G?\XZTH(0B9CY]REY21\;%>O!V'&HE2`D;?L1
MJRA51P1-+4T*?HAI[.JEY<>9@:7E=LH8B$]B^VUO9EUFS3MTD+HR>!5TN2IT
M?^N%X%W0.^;'ZJ%@<"9/B.MAR=M$PO.N,_FN9S^FS/<',O2@J2D0TABS@$?_
M*?9X-0&N\2:?L825N,YO_=PJQ"C;ICXO_NW7/TJ'XSIACF`0M&WER@B]C"`#
M2^7O#G_4VK$0>==UH8!/%_7]I&!N4S'S/\9Y:5[>TRRO?^.D(O[!C#XSRQ?7
M\]?[2Y&TN*;M^`0]\$;\(W'Q6_P_L8AHJE/$MO6*Z3O;%9,N[&!7KF]U:&4K
MS[M8:&B-TIS7CW4X!0T"R#\LB\0N!38ZLRVG_UE5JV\P5YPQ/LG8J_P+H2>/
M;#ZN<_SU/2`&0(+RNMINA^&&_4=6]93]3?;V`2SC$KI5*%[!FK>3BM8_%SA]
M[Q%P\R!K.&M"1=2W.0]<[RS2MU2F#9`'V*%,/P"._BS#@2*^S!A1;V8P>?-X
M:*VM2`,Q_@"ZRSW9G[MC?)8]D%$E#^9C"38F&R`I?>/TF%@4]'`O3*(B^_UZ
M)P7$!N[YA.OSO90V-D?`)U="D^VCP-\':(I*AX!AWMUI8G$.3$TA]NN\UG1/
M8OP+`?U-6>@7&_1[%I8AP%%-1@LYAC.,A^WO/P@"3$/,B@U18.(:\071/N1K
MIP)H2Z>EE7H*YC".YIAT&W2V*S.^Q?12D)<;OP4+CN"J;A_JB"CTFI)X?K/
MZ6%<&G]XV`<33>^\X5M9,2MM03'[JR\ZOU2@P8_+AN@PTX6P@NVP>'7UR`6S
M369#97+5`(AO?PF,#-W>$!V$&\QBA$NJ/0>[6VM+1N#Y!4O="D0.+*FN8&PJ
M"Q]<B=XH#[ECY(F204/H)KL_6LY$SZ#/(F%XNHC#UI4BIFE(L6FXHD4O&$17
MRU(OZU`X/#@\-O;0>;#N0/:"X,44_(-R>28:#-:PM0E1@X8NK!IB2WL'6ZPV
MLM`U0?^E@"I'+(N75M=7+!X`/FS<L@B$4;@15(4,_(%%1"Q:`8)I"+Q0#D19
M&#/],6<:($S2\EIT8_`F1$`^B`2#(@$<V:.?@>RUM$`,E@TJLL_(K!EAI:G&
M`7@PT`G@AF),X;?^#GMW>E3Q:G#>TA:N"/K9<=LN;)$%[(>NN\E7CSW[9@2H
M!=3<-7;"<-37FHLQKL73$T"48>-7PE?JC*"1V!W6N;OA3<('FL!('R=A$5J,
M;O&0>.DPDW$)]W])&*243'#2=*>LMU(U=)+#&$N.ZZB@%__0+(YITN-DTPPM
MH&'^9#J8MY4:_=2#R;'7<)!//Y`5\'/UOV3/EC=B,:!7(6P71+-QUY9V68C5
M0<<@9`P.-.KB0UJ<IA,(DMFN-'N`9(G+OB(&"S:8F2"#*0>1L[E!#.OP@F-^
M)#;^)"].6V1`F]KA/ZUB?J&_O%A5'5M;I^^_IU'_2C2"O_"$$W7>.DI-?-17
MWW^@X,5=MZU'DVR--F0J!_VM$28LUGFLZ6J4ZFP<5CL8IV36S=P=TV_Q:8=A
MV!"G.&L*C#`\Q("9U[W@/QRO5P=3Q$)?#>.(Q_+"P#2@'K"Q610MZ]Q;&:R\
M6:?'(CY?,K&,M>9'L-(?7L`;$WMMZ`W"P`F*LK9G#7_:<0_IM;4@P#Z`_#+T
MK'WNS0QHEQ+3MO^Z^JM@RW^.FOCBZRFRK0F6F+>GXXDG/2LRH(%YMS(".F6J
M3_;`%U85*8'KEK\`&H2?J1CEC7$[?E3-W<O)K>/+A<,?MXZ]1FOA095"P$]*
M[9@L>)V&NT\"B@Y<4U'K+PMW%BH<8ZAU$F%X]@PA-'![.S97`8G%Z:-ROQ5_
M4*%G`[KP_UU2I:$#X0-J$UBJ4WE@,#?7AS2^DZ$<)'J[8UR0$-E;.%B2`_#D
M7`YQ>63E![3JI+FOPGIK;T[!O5VB7T1XD6C.E?I`KNRX(BAI:L0I4"0-K&8W
M/0\?1*SFLG,AP'U(T9^.(KX5<9.'3HS\#XQBL#YKQ%O0^#?/O8PP.>E(`Z
M$/8NKFJ+BWM2[E3Z'\)Z=-SL']0E>'7UFUL,<4Q^"&8"@A1SJOM=7*J/K/\E
M;8@#P=2\51HQ@SOO)JR(.;^(E)D,&C'@?,H"[MJ$ANXP@Y!+^@##@^SBBCT"
M-J$PP*KL`V?`N&\[.B5:CY^C')H-M+<?X*&NR?:^34.'I\)K09F5?C8[G<,R
MY=AJ$#S\HZE7S2C6*Z$Q'$A-L3;B_`$IU(/T:*%H,02_3T>XI<#1,D*,E!BC
MMIJ$'BS\A#9AP1#R=`0_+3\)@CP,':380@9@%1X3(MZ'RZ%?"%SRT1I3JF<Y
M+X7#(3K_X1'5X"FTU_>MD06_WRJF6VT2X;?^@M\'AXI&Z$;.H?-%`I32;Z4V
M&8#]1GJM^H9@2@Q+TQ1$^J9*,4D2OO`ZT,6S<'4D><1WE!EAU]I:?/TG8`D9
M&[:EHZ3]GN,0NM-^!H=%G4C']]?@OVYER3-N,S&NVD`%\[O2Q1O]2PN"4E5@
MD$ZNQMZ]2/$>G'\0^Y$&$_X`7EDML:SE^^\V:LHTR!#=6=XR.'H$?7`TB`S"
MP514@B_-Y/@?A0V.G?-O1F$RZ//<(``>^*"&K/K3,^MLXI,&*\7_R8ZDZX$D
M5*J#N+BE)8YO_S&%N+5[!YU&9S`*\(+:YD]82$HH(A"OYJ`(!-&O`H(&76CP
MH(<-PZ/X[84"GL2=K&@@2EGEF;)7*YQT2$U*KB^WD56^4UKL,(47Y["?%_\'
M4_:*.=L&)SE_T0BH`N1T/HHDD#41AC__N8(%:ZQ1.\;EH[G/,UU<#$\S2$]7
M6J4'NH.?*UM'D<'F%Y6VEJU5_*"H+B23]&]&YIN%(6'^?1C&-1DU-B^^G#PB
MS##BS`GM-9TA+?:P+_C3Z1P2,^!S^Y]:A,,GQ."T(E.YDL"*;GYVW#42'P>C
M-Y9((F<%>^0O,8!D1(I>?N\2&/)6<:,<!]=`B`'"$')9QHZ`^Y\>=*G)2=14
ME+&T5PCA9HRIHZ/6*4S6_%\N@.$8B[$A(4(ISP-LM;`<:O3,UH@[-XCE:(.%
M'W\O!`1:F9F9K5IWF=S<[.V5DJT#M,TNT)J7,1P6XT%GPP#&EZ;/@SO8.(_#
M)QH"AR<N,S.]7WIY]F.LR113^ORL'5X4PIG@$"PD3P</)Q,.+YX+/R]4^%!&
MCG\;IP9[!SR27Q?;>E<'()0U"8_E$%UDD=TQ8N)#G)A"W)/+,GCKT@)'+P>0
MG/ST$USA<%EDMM-`7-P<J6R5C375N@>R1YG)XK[AW^K&"IR6$"%B`CJE8P:(
MLMO[H;(,6/H`NB'N2G#PC/?SM'!X*%*]\R^@:#658<1?SUVDW6A(SR=-C'%S
M_`^'7_?YW=Q5>NJ,/D"BW,^-]_I[C[AKE?O"5#B@"`I8>2ZJ*2"^\>6"(]
MZ.D\'<L8JY!W"&6JF>BI8PT>^P;(K[\R^9E0XXL6&`JRU398]"K$$K1[`0Q
M=^V-![2_UQ*AY3HHKYPDUWA?'_53@#M/SF!PFF:9,Q+]%_LL!_3P$70&;?'[
MNUQV$%G]7"PN(4Y+78PXLY(7:B_N$WC+UE9STTLL$1S:,WE9J1OGWBEDLO$@
MHR@HM0[/^OWGY9MBLF=P\CL,N]_P9]:"<+S@+UY"8P=25L*B5#V]([>QV)R\
MY`1?FANX=HK?L20!L3@MBFBE_T*5F#XEI`RG(PP9X]*CLN_04BIS<[4PT"$V
M0#\H"%[AN(T)Q:/1>@/,&ET,*V#".MA7:8YUY]/X::XP"7/MMW_B:+^A;;=&
MPJT&$_G4B?0'/ZQ)3I;@.,@_WKSH$I-"\O_;9#&AA#T#$0FWU'`A1>^.HDG(
M0:@7[P7"1@')DW52:]=87NB?2;)78LR\AC!"(^5#B"'S(R</#YAHK;J2TTU\
MN54'4(`Q`S,4]KVZ_$V1-3.HZWBD6XWD*?"727HJ-UHUL>B0X<\[J./H[+SO
MBA&"D-5;RISD.AF!Y>G5%!<5`KIQG_AEI9XY.!IB`R=\Q),,/#4AF"-^_JL"
M23?^V278UTC]61H`34$RD@D[\=QQ\M152+SX"UAS4E22>W-D9D6_XJNI,)[S
MY#;=S\JJN]#*EUZ0@Q*]>+7<$1YZ,(HB,@^RW?I"^L;A+:`YW#4^7-^UK@BG
M*F+0)F2&Y/,AI()<MZZ"<%<V7&:`&(C4G&+N8(^P"TH@CM9CNO\MS9H'0HU^

M%<TYXNO!CQ_G32Q@-[0O!HPLMI@(]_>KK*P8@#!?R/"1-%&W=S<=S^<)!RH?
M2P3EBR)RF<TPTPI`6.!H,5SO,B4U),%Y=)/A4(XJ"4^]9N.('XE%=OV1%Y!"
M=G6!36Z'*XH4WH(R+=M+!A>L1)G@L")`QB3Q&L'Q][%36L6<]JV./0HW:;!H
M`-&?O:7,:6H#$MHKQ"CZK?%^RRJ/)L@VS_KX-8)6P<$->E"#)R;J6U0KQHR;
M0.9UN;7"LGK"SQ+XX,-7`Y3(1_%62+>]W-)YV$<*U\;`(DG]F!F`<?_&H_7.
M!A-M-$T%J%5]^S\:@JB,-)`A>N#6\JV%[=_;C>:FFDOAE!D7K>]PB/F(CU:P
M3=0ORXR=A;U1;WGYO6L8]@'A:5<K+=/*HE#EPHKMGU-7OFBDTI+=\WX;@(#A
M[>._FAC(G)B$;P57I2&YN"@GEV]P.MD/N^4!2_09,FCPF`3#WHG!L=+]X.BJ
M&M2#5@_!_6*;.I>"0<]HC/ZKXZ8E#_;P=4]SAVI&KS49NH"Q!=V=S@#R,VEO
M0MC<"!*+XQ[Z[N=*.@`)#EZ6Z"\";06G'F8A./&)V19-BF!U[3_F:6,+S2$/
MJHDP7GR).SR)>KOA8QN!U/\`96_X3QU#:(]9+LVXE@`7Q;+1N_$Q+4&A)H/P
MB*\6]O?-VPLH(=%U)'X]W!(AM(@>B6Y+DPFR`?((9F#ODUCPX`8_YE[Q-WX#
MU#+Y*E?57O1_1;<-S@QW')R1^S<BS"MP<HS_,0_+A!<?6J^>2@PX"[TG2N"N
M9N2D<4**F_!B;XH]O$(W75)"*E2>0`JVC`S1+=&/-H,OZ4$IH.+%'I4<HHPU
MZY#D>)G!]FZQ'`E3U.'Q?M<Z2N;%^V8,F=@ESI4K,(5;#`2A2!8KPH\!Y=0G
M?;2"@\4;":-H0A04/T>FIR/,#6E-D;S\8,^(RC(9'[2X,8U.GZY:AB]NV]$$
MF[@QWE*_5")!.2*R/0,@M(['2+,\3'X_-T(_*C*`Q7,`UAGTN(F^:0BN7`CW
M"19:?'_YB+&(6$=B8%X1K8-X)-N^W$Q)R50`!000T3Z[)9D%ZQ)"A=@=1B0C
MO]RO[V)M1Q*,*7`NHJEH7)E;&E&G+Y:04)SOI38`KS#/W!4FA"I4/4MI9V"$
M=&8D'`Y-0[=,1T!![NP/MB@?=Q%"A)4G[A\FIW3B%G*QCT]4)/`&H$@*BG]U
M['L?=;`Q1EN=&4]PS,&+V9P1I/O>)`W-$!:\P9N_-I$S-(]S(P!(;1>ED8MO
M=SG__"K:[.^;MZGU6Y^S,G5WI(LW^]?_RTTQI)1G>%7U=]=$PMPBJ?"%E%
M,_`P@E/9]6,'?VG?%+0J.,??\;S`9GC[30\Q)<VM=""<&I=V<30Z,X1%2EPN
M?(<[X$06^&!C7_/JF7KJLC,KF+>]IK%F_?TE03[?S[_&7%4=?TIK2%'0$>E^
M1U$0&2"PZ(YUM<@C&8E'1CK'9+>#S%=!R+D'/M\H1;S:B`1A^6Y&KV9G#9ZE
MA&-Y2P6I3KOBRUS'0C)^(_%+_`(>`?X_K"$NYHT!JL(X%"A'M;7FL=Y#7NCF
MNBO+-TZ^\B@.#1_$\X7B3>*ID7[@Q_7@YL`1FRA0$[KF,QSU?K8_VYM\'<M1
M)XKN$:H5RG"=6Z<V_<@21CZY)>(S1/+\YS/>-6$L>U[(-;[G,2_M6[@/83*+
M;_04S]&A+.>*1AR_#%U(C3D:@>%LRNQ<2E!((U?UU)DAN0]NRO-4#@=8&KJ%
MT'<J5#3O$Z`MX7`_*1EN_+.Q1W'T6@>U-`'^F/9_K:?8!67RB8#T;*XC5\;/
M`6I#F)6[1EK3K0VHL9G-GQ>O!J,DO<U,F^9MX/FTSH?18(%=)`$Y"QB]SK$Q
M$8CUI2$B3,>A&NV;T4*0170W9%\'K\@6Q(2^MLRS@5-3D3X7AL)!_Z8L'3TF
M92U]+R_7`K.W1ECQ(%BUL83BD;7AZ5PC\EP1,7!,5(HH]65JU!`>Q,,#P$!D
MH4UQ0B#'V)BX(0`ZR0/($()>?,F7`MB**5G]$T($/L1.B-8J([FQD)0M@E-I
M0S-7>#B`T`08);Q8"95(S:'%?H7]WW_P;_@AM5GOTT!``AGT#,JZI<,+?>S!
MV?P/-2]#]YP6EZKCSL8&L/^[]=ESDYG_/YP+*&R-.O&99)T13],<J2(9<+!/
M54NV5J-PW@_)UQ'Y+S'C9C%$+#'S++0W#1F]R3]XS2E-^>,NZ$.5(+9"MFV3
M"3\EX`/'L5^"4]BHI1HCG[-KK?"C2&>S=ZZ2?>TU<IV=!T;>9_E>X&T"C3Y&
M?"F\>`+XZ]WM79UPJL93*:L;H]H7&27'=W-GIOR1_V&_<J/),1=A,?>!4_`H
M=IC]I//"C<26&?OK(J`1,BXZ':\T>T7.Y3L;ESK^9T71:MUZ5G\5H(R?$/II
M`_<+_,UG_[5F7F:]T[%>@!?]L:^^S:Q&W-S$U,T,3A+R/8(,9V8*`."%NG8G
MDAT2KS(++)V:1M3E91R*`P?Z!T>R6-!?#I&>&HO":8)3E5%E`Q[J63^%H-'K
ME;K/SAC>D[>T_D:R&;W#&J`!P.9EMD<_E^HLC&3OAS="0JD%MC/)$1X,Y8YX
M_^I54>I<NY_N;O.+./='(#PUSJVJ9(3"N`Y4CN6.&^TY'[;^L9>;%;FP*=1H
M:'V-.HT[^L)"Q90'`M'9[R3!NT]>%(XI/+M*@^:4+!;]_:@/NM\,Y$%/'<[$
M&X'DW6K:9$736-KHR(3BP#$>@NR&@E=+><2<1<BU0E.LN$UKL#<%#M_MIX-A
M(Z",;VDS2<J&P851-?(FJXMC*-BPT4<$W-L=^RMC6`C\T9R4VCVXQJ7RR]G\
M9ATGJ$UHU02P0HI1J8-2E@;DL2$;'5*X:1#T!V^_X,M:.)XAA%J:*=)=0@R.
M$R_-"_U+Q[CTPPE.%1@`8`&L-8&<R_VC'2?*2R>U$Y"/3+4SZY'#83R'6T?5
MHDJ/W;&;ENUVZ6E/TG_R*TK")X(-#>O8][^T2"L%A6[CE&'*,A_?<$7NUJT
MGM;GWIMZBJP`;X6]1$KJ#$8YDS<+TXSA`'5`S>WP'0'Z'\OGUV1@KPNS.D.Y
M#]$3E3)3`+X;(:J!V3T.=BAUUGX-KH'5U^^!43@HD79(?-L0Y&>5I54%P/S8
M10^'(@HBWUT"CL"P(K3=@Q<Z?_0>2Q]F"G-@5A97OW`IE'R7Z>).]?"C_*>1
M$L0CR:*DKO-*F&>?GY*`'!J0G36P9`8'B`?0J3:LH00;W:4??7)>&-V]!?:8
MXS>AZQ*WJ#ELF"#LI!NQP8N]?]<H6_+YYNHH>DC@$3*$I.^%*QVLT&07S8'<
MWIP/H,LP>A4'29G#PM"917+IA'[%AJI#X@VBBIL]/K?L-E%?#QS3*\^<BW>
MWDY7Z_\[$H7:Q0S0@N5?WG9(C+F'H^[VH$YWV"GU[4TC1P%0IU=@6U:3VS$7
M^?6Z*>9:#Y,AKGBQEVY@EK&-HM[81XE4D%J]9Z>V&<&?/70!>]A\#B:(O?U5
M0G#P=:S]R,\[KHN]35A#AP>M!8U%B>.S7Z+=>-?-D%=%BOU/0FT$;,WW:9VR
MG^%0"RPM6-\Y(CZ,]IUZ&^,V8[[Y-Q*[_=OXB%`O_)6C=Z?KDC?5CW0E>->1
MT+5-0&S,#:KQ/`OBQ+3Y_AC6QLA`.NMY%RA)&^?B@8!;O#L%K)[E];8@#&DH
MFXS$Y#Z=0V)?6L9_>4[\+^3^<B&Q?9\/2<R^L80DXYPWM)JT'B(7_2APB.;N
M\G[S)/.T=V_X-O"CU";(+CO8J;M(+#1*`(#KOF.S<<\9D\"',@QGZH0OIKP.
MZ/[M/&)$*S./$*"EV(PE(_K%)@/8TUZFQHD3.]G`00^*H&?TV&!U0!C"/_%)
M)GU/5;*N?9T372QG'H$_8&XP;7">7%IT.=W1X-J!?VA>G5)\4/UH;9-LG/*Z
M2;5"<%[/6]^7U(F=\ED$$Y-FHX%9QF*IXI">1[:-3?ITN3M/V?AQL.JA1N/0
M=ZT/)FHBHIU`"5@5&1/?W:S^??*?=(.RLC)?8Z/0T%?F_LBD8%$`(XP(1TWC
M(\"85;!DFL0&Z,.8J;/J/RAQ>0]S'2Y4+@K?+#HZ3XT<.216<&&RR7T$S<R(
M76\D5/.HE]6Q3TZI"=8.E:QTC];NEDMRUM>%\YY#UDQ#LHWF'?0*4Y*<L]!?
MJ2BA#$:4V6<K^6LP&.4YRSKT:BL].'LPJ3)JJ]_H?I,+154\!C^:/>I=3J]>
M/MX_/N(WZ6>O6'2D+_RZ'CT_*P85B#8&^?1,*LS46\3$?<_.9F\^^9#4ACB
M5APM72R7:#0]?7YI2A^.EPP&I')RZ_8X>]V-2>=J*Z5>V9(Z\5=<BPP[OZE0
MVR.^#+4-R4#('S]*%/_GC`^1A@"I`7=M=]O_OM=1V@1S"MO>@!%`50W(*Q0,
M%,&BR6CET(FPU-4$&[Y:VYHX9Q?:-#2XJ`R*,(XJP:#_Y*&96*1I/JHMI?Y>
M2?#7="4E)7:LSV8=%,KIZ`G5&+Z;Z7E**EEJ'<[CKR$.A_5TFL2=U31GS4ZY
MF]3$>4P.VZ8>QYG/AQ[HT$T7S:=?$_1&DU';;I1:EW9T.`9;P>9#PEX.JPO_
MM-T].CSJZER]O0H^.@HNSPD_O:G>??2;J0=)2PV/BG<UGHQ>[)Z5G7*6MY7?
M/C[J?9"7L<Y(+MK(MK4V2=<KPES4(6CU2DZP*J93GM#1Y;#*J'OS^^IJ5387
M*JE*I`,+9D<@2T(Q@*^T=`XR1"[0\(]O0]&8&%<9I,6EN2AQ8_[=K78B0(P$
M.`2`-H'/PAO_X-55DT"S^E"M;&O-J@Y>048Z3#@MD`BHPM$P!0`94FTV9N0`
M;?VQ%RL;ILP\+9*VDTYTRCO:MU"S(D+5>5$Y"X?X\;CM O)Q]=XAU=Q ,[6

MZ,C7M>(R.]-+CE1T\.4N0TY4YX?SQHL'<Q?9/H7):JW143]"$RWJG*5;>FFV
MF2/KG$L<WA[`]?S]P,BKR:X'FFZKVT-I,3-S_=F5CK(R_4S]S,*6QI;91B^#
M`ST_O<RRNGY#NJY$<]+\581OH(3:+=WC4<GG]H^9;*VCJ>9R/A*5/:'9%A
M.CIGFM75/)PS2(G`_/IU!)^&K(:EZVI3D;!YELQ$Y:C&5.5XURK]^EJ>.TCY
M6;%7=`F?CA+P^V'K*/+P+6L:,X"@!_+`X4MH1&SDGA!X8FZ"<T)E,53D)UF3
MMHF8\$ZJ9J='716=8^1M=Y=>1F'1RI\Z#2[.Z8I[%=Z9J*)4"P.OR]75UMQ'
MA;TFC5D>-8D!CJS_6!MLF(V'?AEU5(1=E]\?9CR4ME96F7FW&I%;>SOLV%O;
MZME/&\SUEOD:X<\8F.@]`</KZZ$&^7A@QMA"3V]'SU@OF3UEL2S'O*HPO45Y
MC,GZ7.ZGR>RTIY1<7PV=-?_[(9+KP-;`,GL')7$NCC?5<MTUVJJZ<O\Q^UI0
MJ`V&=M<+T->-:HS1+.7\SDGYDI)-_XL`BD9A'9]2"J/0<[6>F(@U<K"1K6R8
MS>:0D]-YY^)[9BB(B(;(`.T-)4`H;%M>&B%#`*BI%*ID%IN8E-BED#$*]@X/
M%5ID9L!!,M`H1Q&3+Q%>E[V/1=.Z$BSWUD$M&X^[D4LS4S</2P4W<=-<G!5G
MG6=MU555'3\T;TZZ9IJ_#RI6?A`S,6[#Q>7!E2NW0W]]?;L:?''1:"YIQQD9
M^T:-AWVTK+)L97*H;J3L))^.O5;YIZ-:0-S8]'E7I6N9B8-QBG%&1F'&HI^)
M>6EKE4MFLFNVRVX*5H3#ZSGA&7V]9+V?)?IN8MB8]GH;90O3$>73GUY;?2D+
MSPM'=(=@8*@+LM&^D1X:BR-!`P_-U.X5I7>4I<-@K_/R\K2GE#^T_OPS1-J"
MS/7=R@V^MT\Q\?/[8*^D0?H%=O12424<`,%(!=/@"M!GF.1XE)OG*=9EOGY^
MGO5G7CO[#Q2RGNIN\AH-F4L;A5^T_(+H`?WEUHWYD&.>GZM954-;S]3-.@B
M&PITU'E)WK0Y=L)U79VK:EMC:'7LMN9W"P.S.\PZ'(H=],(<9]_YFA3ZF&RC
MF+2,27!\KXLO=<2^]$T;V=H:Z>7HF9EYG1O,'VPTI<SHBL7)RZSJ7=Y_>#37
MWTQ)W=VMKFYK*UIT[3@ZH,],3DTN9%BEC\2/?.,:3E;T'Z97HM-#8'Y'R'W(
M/R1:V)4V(@^!B\$I@U%'P(2%XX3C0*$IO;0W6K%P,-2JK<-%L\'$PL"E@34Y
MD]1A#WW+KJI&Z9!MD_U+8.FES$<Z879&=G)FO?3\"'\1$`@`D=V"<J1"YK=L
MMZQFEP*>T\3MBEY<NZAWCC0V"'<02SO:LVW_9C4O--(+?J7ZN%(_JG-97RF[
M_OBX^[ML[_'6<Y?F<%S/K+Y\;;V^[+<*`<$I`87':5/LKCENG(MS=NE71O!/
MQCE&;)V1DO*T6(BU6;*?EUV-F8&^N8DO3=9T0_WN.7N'2;"CZ^WA`K%^=6?Y
M8]GN;E%&>[GJT.34+^R=D+_2W"\$*^KKRQF%DCD\UQ[P,4*1T'72%C3<;;K6
M)53GV>%8:?ZK426S0&I*LF?/?U%.O5$M[)F:W*UT*`>K3;U17L&:=KSZ^1Z,
M\>7=O_?N"2&MOM<WYQ\^>"K:#[;8V_NV\$K6TN\,%SX3PQQY-/6914!N_7=O
M=S<.L.7:7M@[>#I,'@@_V]@3CC,N*TL.:8;1!':&F!FL/E15VG?\9G&(EF?_
MC^W@<KKI\?SRWITGSCI#FW4E?_)Z>N#'&MNB/M4Z'%>P+&DO1=@RFQ\^R\'U
MLJ*ZG(>+3%OVSYC?,W!Y;;E31?/*X)=2&YWAV.F4(*_D'%UI.7]=228Y0>EH
M+0Q6:^51#`6-*>.BG-J00%DD.17-'=IUV!DE@;"?")=&V>?]2\=O=0>"!06M
MP@2-PV_VLY(Y5'M^,/ZLKJPNFSHN57M[%PDD!_F2;711OE1WUBBSSG$9,[)O
M61P+_VQ[A_&278ZT]%-+O35]'W\!^%WA"S):_LV/'52M;V;;MFO^99N89KSO
MC%,N?&CP]E8&D;M4KVXD/[IVR^F7RGS!8EPE<4C-22DU+BN)J,R9OP-TRXVG
MV!]^2\W4UH'EOYE%8ZEF/K^\S"//Q3_&-:(;*?I`GVYM!H3\(9$0K"-T(!@;
M+^L<6HS=EV0!V5O;A85/`\F^/QG8L/^^?B9+&))TRJ^/^&V-3?H?D\@G[*1G
M27M8>V^3LJ]D`%D91J<L'T-*0P=A1KV'O?I11Y%1A]K.LOD_AVVW:K\YT32"
M1W1TN53"LTD4!@3KE]K$!:L`H?*FP5\#=MEESLD-C47GW@T-N4[2EY-$F*U.
MADA$?U\M.Q$Q))P6^??=P9%>5?++L*$4D4*"F-C!^-2_?6#%SGU0E,5E<IE
M8@/.47\AQ#!]4MY@#!)#DIG,:5?CU)G^8\ZSFZ']*5C6Y-*MZN[0'OO0PM&X
MUCC/(".V6L.0[!W-P3#GR`\TZK*N_.N*5U:5%6(+AE_ZK^?GC?0*C:9IB;PH
MWYV&R4SGD*K_ZDZVMYXW\M)S-O`LP0C-MIJ,-S'1IQ'3[(0-CZN-]^5\P:?&
MXV8\=53C;8B,^&22[7FP4[K8VTMDK/KM*-O"V:*HK,@F/1W%JV_MBG^1W9JI
MGS&)%9F-KZ;YTS7P\;[$^6!ZS4NML''<<1(!2XHGH++ZHHI%!L__3"48L'WY
MRGP*>4S!1&5(7TFB8EK_F>_RTK#6X/"'2TZJ:P+#.![OO:CPQ;=U<P0_@[\
MX5.I5U%+:7I1D;=+6C(JW\K0@=#XKNVF[=:LJ[JI@6=[I=I\-3>ES*!0O\6W
MM<4WL/']0&,/SDG+5>%T*CDFH-(K%)!Z4OO:4Q.PSNNF&*A0/VL!F+&'&]U
MS8"BDM+*C8VD@#^+AM)"0D/#7]^JVK&Y?7<?YQ0E7&#7$/`\?"O]X:-T9$
M_J?>JVIL8Y]GXO)A:W9Z</#[]4J*/::OF;WCPF3K^R;CI%%5,L%J$DQ,#-D;
MY/<Y$="[GKC_&KC>.%8Y.'=T%EV04B?[5%R873(2K:=0V*27\K]GDP!D*)@9
MB>/_*:K,+%4UEFJ8-<2DQ=1#`W-G9!:8[L_OO?0S"E5F+`G!>2&O"S)[MK*G
MKQ1.'PVF/"I'%T5$1M_YL-DZ.J;:%ONOY/;IE$D&/JSL`V5C584&^NO(M]2[
MW?M6&(1D26=Q8[BYVK-<9-/*AT9Z/?=&=C65_,.]O^R\W8?N]N/>FQ<0)?&0
M3!=:+\+Z@2S>9U,"L>GI&$VD6KK/R.SL7SF.C[[RJ-1Z/'S<?2QOK]2I']9<
MZC[#5!V5?3LS/DF?V9SNNYB98Y&A5P87*YJY#`HRS\4_'-UJ,KK9#;(W?DVB
MP3^Y=,X=:'P_?%DM_<4DN`#:];99"1T+CXJEXE$!EUL".L;()##'9.,:37;
MV$%G)5`)U5/0@Y+HHK/D6.?,N.Z\GO6BB-LL7[NN?NP2J351ST8;U0"85)N,
MSY-8A1H88OOOBHX/2>K5/-K4=%[Q*_L[C:\,?1[*RZ5FTA8C%^0Y_7N:W)]8
MJ7.R%E)F2ORRM6C3!N='IULP3T7(*U^Z`<FA3V'\'8P0M!#V,..T!&P('AZU
M5,!,9U.P_[=5"V<=515SK\VMIL7%E%=@77D>XRX>BVJMZ;S^82Z-NJ']JAFC
M`O2MKH:NAH;AX0,VN@D(2F)5BG*VL?%@3S?7W%1;6PZST"$YU\O6)GCU'A0:
M>VWPN.Q`Z(RK^U?10R>:WZZQL"R16BQ?-D^G`AHE=A+K[3T-FN:/#[C+9KI7
M;599N+BIJ[I;*N?2[V*?C#HMOZD)DC0M$RQ5'A#"T+63[Z<%9_'>&14PXQ_K
M#Q),Z(DTO0SO^66R4(IE_=K$6OM]@]9XDV9MU^-4W!C5JO:B'V<UJ5*_[ED<
MOX-=R6VY09I9YH'0.$8:DI$1C)MG6[ZL5UEJ;)9Z<-EPZ&">D:/RD<DJ.<$!
MS#G27N_X11:3[=X_62D5G:1_QF^ZU,2[_,9LG.KWJ9J&D+/$LI"`QINJC]X9
MA=XI-M%I%!E;E1$57'I"N;A:IR??&`27.ID<3B*_9KKF4)A@CA$ZK3AX'$A
M`4%Z>BX(+E9V^;PJ&"O`[^:O8.7%:<71\7:]X*"JJN%ZZE82TZNSG.'*V;W9
M\C]=#LO!W_@N*:B3_ANK+=YB+/?74_46M"YA!1/3QN`(GG^MRK@H+,W0-B(\
M]K^UPP!HD-2E@"4M?7D)?^ZU?DJ2(7+][.QTO[[MM,[7WV5%SU'_\$IB5_(
MNMPEVUQ_9F5M,1*$AP[#\`JKH!YP3?UA;9Q7H*]K&*^S(@4+P7SVJ_J@^L#Q
MWB1@7'6LO42FMH!155DGX.QW/@L6(2`+$+(R4K*0H:_#]"7*N<N/M/:T;G[Y
MU9'&LM_E*ONK;1I>`XW[?ZNY/:;R?#J9%G+UZOJNBBUJ4(O+,M:\0Z'JQA:
MS7Z2'2(B0/9X"F9CH-;W($2]F\93^+BOXB<F#8XE-AKCGG/[)R2\]:]R-UI>
M7EB4E+S^4N@9,8%!)%"9ZG#VMI563X:*)M8)TR9+3IPY[HKV/=PI>5WZ?IG
MX4)O/?NBI)KG%&I#OZ>AAIY<0NUA/]0T/"8FK>HC8?V6U[U+*?0!QG1UQ:F^
M74X5!0'+Z@J_D;I$"$!9F55`HB(/*Z>C6+']/:\,R7>ZM[3*QG;/]-J.%\4#
M'8_;V76Q>OGU,;?U=OPJ2]FZECNSN/7*"KA>>J=\))-MY@B^&!L_NQ7R#Q@6
MK)FH'.\P/=GC,)[2X-!IF.GPT!KJ*@N>J#GYKQRO]=1#NB7]FLPA=FS8:PIO

M2&70[>_6MUZ-F-2]A=N198>&\3CB/698?_\RB'_+Y!-[13G>U(]O(6%6Y0M3
MQ.35QFZS$1W<1[]$#S*);]OG(MP!-"Z_0CO[7)EQ,^E[U<;]CLZ9HX/;/)HW
MOZ>G[A^PG+D$3\MET'4F'TS7"I%+D^TS1CXZ80P.NV5#P_S?<=\?B4)?,VDY
MCK+*9QDRLD?2XI4KA':.U*!MA8^=K?14:^'B,_%LNE"TK,OSD*M$*G[<(&5D
M_YKW+.P=O7WR[K+N[D$B+M5PQF[&1:.1]>1Z(3G]K!&'GGK3F[ONPA'W/8VF
MLJ5BD@#.>T=*2D-Q]>IWC.S_+7T#?Z(O0RNWZQDBZP/>?#4W]V[5;\B3<#!A
M<DC?$R>9#M372[']L:"*2_F50NK5]4R*C77-RL2&K;5YLO^Y>6P(0K/C513\
M#6G\+RP=;1'Z/_??5O5TW@KHW1=.ZZ7X%187.Y;B\]P'95=OS>YM&A@[I!\N
M^"5L"I5-D>6H#G-)=%0-?EQ*"_PK>STKKB$SH"P7<6NBQCY[M8\]26FL_0.B
M'6L`-]0O+-TXOZ;"[?.`_`0A6V&@D-?Q"2#!$0_BQ$IU@8$23VW`\KOARHA,
M007Z^*ILP1OCL_.I"N5H]C8-CZ&H'URWY^4SX^*Z54W3(^<-W_F?<>D.Z0WI
M8%-D,1>@F:+)DKP&R?/JA.B""`2N0(3:4`6<*(_(R(9HZ<:;QL7DQ6G;#17Z
MAGF=N(W,N.1JZZ_XHVO8T4O4AK\II)8//OB;'UY^J>U\\T9QMX#V-4U\6R51
MF\->U>W18V=G6>?O2L$IG.>/DC-SD,\?'/0VUCPSP4.U-@LQ+SVM;A\^]1FX
MI':NLE>,U+X8?:4?R0F8!(*MIECO_J2-'S]P_.0_VLU(MI@)]%.)0M!9'!\[
M&-G<'BG+;$LU?J1@3_D.]\!'T["I$:955JE1QL#0[]VCPG8+HX5!1G5Z;?_S
M/&VJ*FIEV[C>/$_>3<4_:6A;=78Q][[VOBB\V-SS?DBI<</?V!S20RO&U$5[
M+6-$,X#[)7YBCG$=K8AG'>,N#2Q/,]5.T;AP\>W:M=R8K#'EW9&>OGQY67G^
ME2GERVF)!G'USQ6+$N,NBU]W,)7EXC-2'2EQ'?%S(Q+?S6IH49EH,$!4I\S>
M1:'YO07(;&I&9'I/HF4(=>4L:<DH#;OS:<7^+B6[=QC<B^T/+;I&W:0FR:04
MZH_OF&M\+-)SE%S0ZS*1`4BVKP$TDM+UOJWI/9,C_8;`V--1[H9(![U2_18`
MIV,L!2^[51,W[XOZ.:A,[]_T`SPWGO`9%/0:A`DC_9X_(:S)&S"^<C)RLMLX
M'OLWV'QZ&L9E5ID2_V#OB/ZJ6`=_5__R8F]WQ][F]8IJ0:6,"0?E7"?<'P4F
M\].#^BQQR2_%F/EP6_3&?ZEC9D2_.2\JGK3,]IXVWO7VO<<_S'5VODU&O=FY
MF>FH_XU_)%M'RTT0GR/Y3[)_BQ`4<WXF':HC\UM6;.B_W((9A!B"'(&/-<%,
MG_1GO9IO<^O/2#EWX.2Z",5SWE=_^KW-?&CF,$TXYO2#$D/4D)%-WXS\C+RK
MBLT+7;4]H7F-^O/6TQ;WP???,*EUE5R.VGVI7!2/+M*H:T)<$$86":2!Q3R
M)2Y&D+>%BP[R;#NUV#'5)$7?BS5-U2RG*)H=@QY,9&YOEI,EG;KG[RI_TW1Q
ML;6Y-9>6OUE:YN7B9:_VKFWYT7_ZWR[);=Q[NV8G%=*,ZC^.WRFMA`1KA_KB
M&E1_UW%Q2;2W^7VGXV[04:L]D-:==.)T4HD=(=?6?@5@3'2SH6GF-*82AB#3
M3'T<'U97?'"*BT>UWGW&^!LNI$ND#?!,O#%OZ!,86KO>)6&?OT[4G]:O0__:
MV"S=[%)66%;+\"V1^G11^O1OD'N-J*A/\T/P]J-NQH_IQTYY62.7P,LMNV__
MJFK-7EIV@$H^1V!H8Q&':Z.I_$11V\\J8PB^*\2)$V9A",A*]'`\&;##IOOR
M_4LX3GO%$DW=[NEC;75M$^-@F<E.=653U-;".5>E4`,?CTN%L?>C]&J4">O4
MY?U]H"E^AWRA0_'6^75G:<DG)E+%)-\(C09%D66',-@UG6EY@TZ92D#MT.^A
MAC=J.FU$(D'UP^RMO\MPL-V36H=UAL"117/KN/CJ]O#X$B)&,"*FN-=1(WP"
M+A2O_9:"0B4<_AS.3$[[?L7RF!;9BZ:.48%G!T1;C;XO??V:5:?J4UGUO<&]
MZW10^4-96U5KV^YMV6ZT=-063T.7),?,^21/YUW%I!@V2*\J`=`#2M"TT6WT
MV"=(-(5"PK<2-#6UPQ-Q-43*==ALVT7%[$[_,E>:9J4UO#?1V]@>%*-%(8AP
MB+!S0V2&#@[V-)SGKX;?Y([LSM_G<-W^G+H1]MI,MIY9F$[58WRYD*Q2#A=1
M*=PHK$PV3DZ*X%DLW`F00(!?^U;EZ">76GT_4)Y8OR*M"GB7%PD[@8G^`>A#
MZE%#)`#)+XZ2`(?D/U^P,]H;)#GK6_/]IDVJ*_?N'CL/V[B5E[X/$+/"O\=[
M8$>)RA=C1%CF92/@EBR&2R)$8H883"PR2#`&/8"%"8$01Q<P"I90[,_J/?P[
M?\T3<M[W0D]#3TBW0;VA2YYB4I?B%]'[%A`SD0Z1;5AWI(R"C&W8G_>$[@QF
M(6R*&WZK^B9VAL;P_/PMZ$5@Q_7YI5[GTL<CULWU_\(RR(@G!WOBBK>:"K<N
MBXNOJ<[B@_W1KV%A_E?!6KN[>W+(S^\M#+C9Z=3`E1P<'4J%WC-3KYUC+V)?
M7^J3R]N^2-8;Z1#\^'(:[I&?O_-/LIQ=;%KY;->T@;$$29\'M`IU6%BD%B8S
MSSSJ&VE?[>[J,^#/)./`HM'*8"]5WUMM1A=3$Y7T*'#[DP&.]_%Y<`(R]-#M
MW\29L368!/)M;U]ILP<(9O65G4FWQ^$\K-X>H[L:DMFY8$&V)F@Q#`F4S5<F
M).ZQ,<<G$$B$#"(O")\W=CRL51CP$0%Z6[AILQXQQB=D)I`R[8]C,7/X^",(
M7HE?Q\(0CW??3`4D!3Z(UED(G,(9)1@Y1"%-@2WGN:@5M$B#S<:WP%:1JW6D
MXN&5#UMHJ"AGZ.DI3TCM-Y$/DI8ME9GDQL\SR2.ZAYYGTIJ^6#.RSFE9S?1]
MEPG*)@0MO3!,34'27A.\Z/9/G@^)%]M+:&GY2-Z0Z[N*4GC'?%2=>(RJAPSM
MSP[#31>'G091JOUW3M#M&A3W.%PJ-B?V,_;H:FBW;&NP>YT$PD_$3YB7)60O
M]*?0'4R]`Q_C6*2[5@$_3PNW/38F:LPE.CY`%7NG'R6X_:=_E2Y&1!@""0#A
MS([L1X_>S:*Z\5$K6*BP`=JQV,^IYK':.(9WDE>0[Y"(&O%B>2G0B1L'U97P
M#5TK+`"$1[GOC(F:1]6C)K>_C%)=@E&#PQ(A:Q5X9\+\-!$\WS!PI!G?NS(@
MN]W#BCX@%`A$I"'M(!SCP8`:U!^4V)KS.&<%]AI\W#R[$XJB^GHL+!(@B9X'
M-RKNO<%X1HH=D<5L)BY)'.@^__U770I=MFB+68A/_S/B./*-C#LN,ZPLB;P@
M()(L$AP.L6Y6UA861JT^QU-$]]+[IV)[!&M6"OF.O,^!F]*0WT`I<])K\2=^
M14KP+9FT,]"M:`*&I'YGXK8_L>)`53Y19V2>4B%A4$3X%4XV.GEW=U?W6OH2
M)8B<;_9'!IDG%1[QM"?QE[5`-FK2;;QW\!S[S55%I+Y!A&<!^M]+V)B3+RD5
M<SY\[GT:TERM*>(:\>+EM58"/$^\D_H_?.S;JL<P#W1$\ZT[!D<Y%?[G+&EA
MOF_(FK$8!'3$1?BARSP,H.4]&'N!1SX?CVIK%[ES/*PWMEU$I,XBE(E7&=#_
M@"2OT"WHXLRA$N8*V.^D[@#Y;PY7G7=`3A:$@54&]YTO[,A.V$?TIR3CN6QV
MF-5XQ=S?S0/"8\<MV*L!8^(Y)SS]^+(<?"+_8[Q;[,?]/1`9Q06*8N.<Y.XS
M]!).[/N>%8V2@_&BC?$\B298S#)L=<5VML0N<3$Q7OS$<W'.F)EX]DOJ*L)X
M"?R`?@GSN@^1G(-X?Z\.`A79KB*4@]Z:$R\BK*\PP,XN8[V(G,Q5J=Z.!/+F
MV3;6@U^4BH%ST*@5+>LBL(TQ6F@$TF`%]Z_F/LF_>]BX<VX0B(8P?_YIDB`&
M&@^?!\;*RN$_?-4NP@-S1>^J-!:/5A'.66K:1K-1DG,PW6)#IA\(!EXW,IJW
M%]S;P-JN,)`F)-/H)T/5N<_?/[XB&7$"K+RF8;'@Q?VSSM,$\!TAR3`X0Z)=
ME52"0"0Y>)(?KJW."=1#>7$S96YD^`Z0HF*C_L<0ZB7PJ20OTU'O\Y,-GL^+
M">(\%<+@=FG7,>HE2!I2K[V=I8UEG?!RE2FJF-)CSCH:?"R\&']U_7&718D5
MY9XZM[J^^'P5P;*1N(R`4*"G52?(T"J2<=X>0Q:#EL#N0]UM]:=<S9,&1^D^
M#Q]7AK!\@[\T":+@CD#B]7GV`_J[:J.:U-F!V-Q44*QO:O!=^8U??7H->]ON
M==9R?Q2+8"VO"7='SK36:@A3=!222&D[C0\YZ^U*^%MB/.'`^<02D.E)>OS
MRF^A+QBDED!5<\Z!C_R&N!&U_L?H_'X!;7UK%NYDVK1SE#11F]0^A-_7**YP
M_@L."$@;(97T5:1:]O[#'S7>J8G7<04./F:BT/T#?T91ZOG8%2NAHR(LG7Z4
MKT[,Q?0:3<CL]ACF=W.+UR+WBM6];A%AL;I584E_YW&$I)=Y<7/RGR,Z]1VD
MXNN\R<^?)T0QV1?Y9`=-&OR!BRO*$AW-8*\3CI?7!6`YS`2<6TV/27O5'Z9Q
M'\6O$IM2=ML AZ>/S%^B"&-GM8)`I(3,P;-"TG]=X([?CWGQ^W%:;X^QS?^^

M_8`^$W@%NBZ((;K"O8D-]3\&?)N&?WY\.F3)+G0`)73L!A@-?(Q_S=,%[(D<
M(7D^2Z,#!_^+[M?<-H\3H,..7<(&W_RBJ[Z^S+ZCWH?D?>?I_A=&(-2_)980
M_Y1W$,XG^ED$4GV(S)DFZL:/)>L%I@2.W:!MPPE`P#J45AUNG'Y%!AB9`SX&
MT7.K7<`!JZ4F(^,7*WZ/`74,)-$U+\U1[='1/66X_8CK*@+/L4;HQD59S16V
M8XFBA:+55*,XOH#7W_H=0BS0`7H*#@!;+7Z1GKEO&-GRM(8]%SJ\O_*TP7@,
M%A"`]?M>LA%+>XTFX?GEZB9XI30(!H18VH5TAUHBTDAA=&\MPA`^UEL-IQAM
MU5Q<=*"6"E[IX`^1U6GHTD'`FTBM($!FV=D'$G)&P+8X^N.*,.-#B"%H'(5Z
M\1:XZW<Y[MW<0P[W[?%T(['WD&1D`T@L3MR;8S`O3JV_1@B-;6SU7A+^IO^A
MT_&<#&UI([9V+1ZH.P&N]2Y\0!\TD\71^``[<KH[QO+'C81*"?_%%UNT(25V
M<&1(J4FZQ=W[^3#<9[4N+?6JR]'JK4*6"UO..U_<&C`+90*-79!U$\;-_<F_
MT]NX13TO5^:[G"40Q>.((D8O;@U>;0V^^`#.[?#PX3_\B.FPVMKE[66:S.:Y
MX2,D@5@L+Q2V\]A?4\W2!=R-$YP823;S"YN=Y*Q6_GW"19`A9B3(3@QQTT77
MAV-SXK_TJ]TI3^=I&QI;4I$\D'<S!\LN/%=(\2.D($T*??]_.KSX;V_=H'_\
M<?_KN3W&@\=8K?_]WTW!&>]R/$,T$U;-Q6TL5C:#UF,@^!#N4?'X\2Y`:$\8
M^PA)',^\W^`WN(>4D.!Y)#("]YZ<CN3<`7I,1/&Q_T5U`2*AG5!P2L.AP,W3
MO&.W-.C`.@*3&H]HS/P.!*ER^MD&,`!UA*$40J'&OJ2S.6"TYN!PR0''\\)D
M!&XFSAHE\2I"*`_CYN&*&)/&/4I=IO3NZ908\7+1`1GT$$#H&.#+M0$O\3[5
M<WP[W(3(=)3L;`?D(!G#*`%>G&B,Y,FG7OG[5`<#FPP_$D>(`SD+^Z(.D7#"
M7>]'WJT+'2&E;G%OCT'*V(Y><U>B:)&@),`9CV"0#O,/Y]<['/!MU3&O#XP"
M#GUM=Q&6X]H>;D.@7L/2])H3B2X>$7OI$QM[>PP>*8XK6CG9X)>)R>KLA<>3
MU33MWW7=;FH2YL^B4-`+^/.KW-P@F!0IT&0I3%5P.3^2%3W0=Z'+W'&%3F-O
MB(88R>TA71V!MJ'Q\^*!_47"GOIT73R`"G#)#V(Y'.Y7A]BLY=R!)T`QP+_R
MJ([\$+^F0<9,\@\;/T8BPNCJRUKY\`5WFCM%O15G`X?@5,_@-/%+,]F7[U`H
MJJ3S@9&/`TL4WIL`2)G<\$I,$281)]9^TA<8B%#%$^''^,5;`0*%W4HX0)AM
M`+'X.<I.?TOLFK=;A`C;+I8YK\]7S?UG5$\H8XCVSQ!PO7X4P7I<3/B>*,US
M0O\.*^<1>7NR"%3BL>JOI9U7,;$BJ6^N*)M)2%=9D?34$(\1_!C>0TY0V.=J
M/*4?:9`/M+&I!GJ]P[&M\^"'\5C`]A@5YG'E*JTJ@H@[HN6+[E_MDTI)M8]U
M*BH^M'A7W_PBSIU(9CB`HXQNT*&^U8*5=>H?DEP?UY8>"3`L'`2.?<Z[1*%
ML613!"!MX#1OD>OB?(V_*D.JA=R+<Y;`MZ8P6'30`QX0_!\5G/1Y!C:RI]GD
M!P?`B12H.#T]D'F,Q\X<MH"8WT'BE1Z"5N+!'-,6QSE_X7N<2`RLHSYX]B"(
M%H,Y748#(F(4!`.1M@4H`A9!D`LD)4U:CM6%RM=]MAK=A9M+%+E\5,L![VS@
MN`TH(+;)G+_N4!VB:;==W)2$E/AU@&VSSIA5@R<`08,0Z8/*^S'MK:]$P4,J
MY"J&:"=N&+Q\'XLQ0KH`>=GUAK^;KFT@V_K9AQHG)/""%P>F5*K1%X]G`W*V
MFV,4OQ)%IQD=W42+2/PGA0?,CRL@K14V=II6$<$OGVH8"G@;#^A`(@'QRI\+
M1R@48;J:.(&:],EU'3"F4MUUV*GTI(]DE5JMT\"^)*625S!_+1&RN(]H@W\6
M16ZED%@5A@3_:P,:LC2TPO7PU14YT),,"7C2(*;W6Q:XVR*\N/#H`\+S\S&S
MKX80J*-*8!K8@OEI7P2"1)(4NPGG;."@>E.\;&'H)8H:#(S?M@]G\ATBI8X6
M)X8/ET9O:P#+?=IYVK=8)T])6JWFV.9,B3RRH-WHV^\O4"`R2L9PU=`HQ/+K
ML%E184Q>#(5#P5H9BY!C[XAF5"AOS?=.6#ZM3U#@O^>&\+=6;TU,R'C%ZA9_
MO37WYG^NQ89P4[;8*CO1O][/\G7KSA:)OA8/:7W"E97/&#(M=U207LG%N''(
M[:2PQW5F/`[ERUTDP7@=:G/GV`V+%^N9&#7&%2<CN71H>A?W5>IK/)08MD=4
M$T!5VDV?]L2!/U$I4399R`&2#ZA6@GP\(3%U(/+E+9!S!YFK`X:;'/%SU'3N
MJ`"ZGA;.&$!IJ3=D,8#"B;SO$H6[O)X&\)XP<%65C]]'X0LU/.:^0,3S+R#)
M-*J3LF<E[>"608/\)>+7]U0QH(5NIYS)CO8G$--$/\4!(W-^+6Z!ZL*NE`9>
MI/^A0=77=,!K]E$_3$C.<\,0)E=DZ,;0OX<)_.,QWMOW,ED]0O1'BJ"?2/C/
MDL%7]I8(&L,K42B&FEOMGYEQW8O`*ZA,;B=<'TY@-I+KB$(1Y[AJ"4#OKFF-
M=[/33(N&M&OA5+;_?7"!?831>_=TC]<_&E8T@M,Z_8^A&(EXO?<"PCQ']5LI
MF3KX/"$(]_<UD<;`,Z;Y4QM03,ML2$S)U-CDZ;)0/.%J$^#$Z@10\4)NDW1Y
MWQ^[X6CVS7@=(5_XNV%C;WQ^,DK,(-3\V?6WX3C[ZM6[U>C@&J>75Z(PDR=P
M>:WV$6'"9`*K,P///?T3.N,L3.,(>"7>.XTN6<LHE&KD7`O_JP,<%_=O[U`^
M_L>XA]);3T32:92RM8I@%/BT7]3I@WZ*Y_\1\?Z_:+$K40C+<FK2$B79%V*G
M-`BC4E#TY5WJ?OWKL$A0Q=8BPMRT^,&L)YJL#Y@2;@2]LXT37T]<8M;ZUP$S
MA'T^%K^(]B\0GG3FD99@@0+*3L9+J\3DFFAYADUT3>[>)"486@JS30`V;4_[
M\9.N8NFHXP#&GRA8B_#;^ZXP!/J54WH:L`P[.5]=,33]`D=K)J<E8S@EDXH9
MR8M%.GL.<%ZA9=)Z/T(_,@/*$\:$\&,,MZ9?V!Z<0)-V#B\GQ<U-YQ$!-RT!
M)B_OJB+<*JB=KA7W;S@28BH#>N]#^+'ODI+<+":ZFTSK@]/344+L=IYD'9E"
MX%;4ZDY*:N[(<YCD%9@?""!!4-U?@?`'^*)OCW"XU@EN3TY[^Z^2%F1('OD]
ML1<N42\:R95&`^?0`SF_K"*S8Y:O<[?4!IM>>6)<;VGSF%P90F<,8F#HG3^G
MX=0'@9N38F;MU_!->9E&-37&71"B#PF[)%J/R>T+$]?\L")YX1LX!UYNC:A,
MI_U^%368HQJ$QT4<>1JU?TH;`PWVI9VU_#E^]6X5P6N\.V=;>XIYK6Q;A!LF
MWH(P,82S==[67,WCW+E6<C5>)Y%>(IL<%(%^Q/(@W`/^/.?YM7V/H*86V+2Q
M$=M&3<Y_Z#^A=JE?AGP9'L8]\<%"4J>%9N^NOG/KEWE&8J/ENH"NW\()GA,%
MD$+E3+<RM#99VW:(9\:]0>[9'<YP*O+K#[MG1UW`E88\<_7GHWV!=42E?,Z]
M3D&O]+GM5BXEC1VU9^8:-%3S"2E303CU*:!G*=8O!`9'TQ"7=XJ@P:L0WR,$
M%?./J%Q:9UYA#\S!7J`3KON&7LM0!(7"_`X?1#@/K8K#<?LU#752,#4V5UW`
M`6M)BP+H-P0KYH;*IF(M<MC%UV>J$OQPV&;*8-$2"SO`1217P_(26TSA_^A
M"R!([T6A2)^3ZQ`Z``N.?^!4?RW&VQ].^!,Y(9E15T_H`,24I`#SA/\DMW[\
M>\+GD.#[%2%>82CSVC?DT_["1B+1;/R5G3SGGHL"3JC`J*Y5H.HQ1G2T@A<7
M2]:XB2AA?;E@^2<#`]##"ZJX\-;:M%CY`"@WD5^B*"26.[*_2NW5!-0S6PH\
M]59#2\WZ<!<07G.M><;,_]DC&`(P$O/\]]36M4^""CN^AM&EFI.K^?*>LC=^
MY$3<$T<>7@=<W[%T/F_OPB`1Q8ZL)9`=214%#:-0H-0I($\14+D,MBU8*GF.
M@H-&<&DIY]PN46CH5M2OX.37`:/3EG2HN>[1HJP!6K"F]RWPY.)G<!'O*-*>
M6"\B;Z&K?<B)$2GVGW+X6W4`"OOUI-R?B\3\Z90>_BO_#7"!RPLG1+UC<.<G
MLV.)]?TLBL1"8L*D1&KM#)C=OB+*UZUL%'C%R,25H-Q9="76B&!"!,MYC0C!
MVG63"?@;'7I)=5UA<"XE2>U]B&4B421NDJ34OS9%(*:Q\1H1J`XGXN4`-N%\
M_)_;_O#"<E'2@X[[E\'MXP27#/3+H0NCXUV@P"O@#092$5:71+1/NZZNK(,;
M`'7$_.C]O'&&M+4I`RM@N>*M)-8W\SOP;87'\\=@JTY9YG4Z1]022QNPCZIC
M1!*G]=GA%<!AL!^%"%32?[=X9ZY(CF$X`X!)R+9B7.J!+W,M[INM#DH-^*NT
MB(,_6(O;\>;"\M*T%Y,E@U.`JI@IZJUUWC"BV]DI`R:RE)/PR63Z5"KMC4<F
MXGV.[5<MQK`@4N#?)0K3,M@R9SP(2'7XS_X8)27-F_#@AG-"K/`;X;B4:8UI
M]:O!GC4?KO]Z;8HY"ZF^P9COFA8+]28,PA=A(4>XT[+(&&)<=>$#IX176K`;

M<HK*!SKF8B<CA&9F&MH)B/M)'0$J"#_CY^0J@5<0(S)T]#:$X[:R.7FZHF^<
M-V""@&!D`Y2)][^19$+IO-A0D\`YY(L`*I!3J5@B\%^\:S=28%XDZ?`=R[N$
MAW5/)+IOLU8S)]4(A)U/`8K(%=ST^T#N]#HP]_+.%86"/5ESCZ5_?(DSUL]7
M$7!#918\`^RC&B:4J/\Q2'-[9^SWCI2RH;%&WU<?<(85)2_VSN$(_GC=R.8N
M_;H;/;HADO3W^?B?4&?<I_PN8^OD/'=2!R^OR27%8D8@,4M(\,]3SAB(!#LV
M+YX:]K!-[3M]+0Y@G\H[.8FDA)443I5Z_>'B.?.3!Z4(X>?8671<11A=&:+9
M@*YRLI;O.?HPQ)1#!YO[YXO)-E97$6_PS<@5X@:/D+Q$)T_63:RR,HU<#!J(
M(BK:!,B[=T8H@5%ICX%(RN>I&BH_FD&!-U$TN<?N)Q27*)@A0,U/@L#N.>DU
MCQK!P(!3<'(A"5LDF\7E?39XL8?'`5?4<.?S\U.C8MYF=SA5T13H^I(#?Q=.
MWO"56H`3,G85`XCT1B5"1I1F2/L*U0!CF$5NCMN?*]\F"<H'YS`_OONC54[Q
M/B=D8R-)5DS$S])-!CSA/TV'M<L,%=Q7IR>'QD>=JT^5@T!R"]W>&X^(3-.K
MP8HLD1C^SU!'Q\,L&Y"/-YP*3O[!;R;5/OFBM8,77HCD?CDDRN:`.19KO_1S
MOG=[^A)%)"SC^;>WQGQ<DS`4701XZT85A`=*#8(,!CBEH_10:OC\QFY23T:/
M$(%UC??W",FPV2I]SP0W(,IY/S^JM#L3\J1#Z7X<>NJX&:<KR+_/5>YQ<?\S
M[:ET1TBKBG"`/_\<TL,<FHD>@\3!G3\:JQGB2J*E$?XH1856"`M&N_FP$1;"
M&\F8F08B+WM1$`:4,#%9(JA.>&F9F55[:\-S]R>)5C8\7E?;BH**Z?U-+,\B
M<>!S>]`N`1H((-/'TD7I6L-TCSL"OJ([:/G("R(@3P.AX9-;+CNE!D'YR2Y3
MOPO@VG`#(OJP/DMAA09$,`5@(H1XDNM@;0:(2$?G-4+$A^%)MEE9\=[1>+]E
MKXT!-RT])#\$!7F10=!,NO_SE)I+@[[2A\-^_RZDPQS[,C8KBG[T&/TO(>$3
M)Q76<L]XSU4%!$2)U"0BPS**WS$M_W"KV?GNP6J9UVT(QQ!)QIE$[@/B)Q0>
MZ!:V1KA9$"+@+,$LE%<^0)Q)3`UV7ZU5ZAVS>R9:6SA@=4M96!S,@)[G[$X.
M1TL79?9$O"?#0$CP]V"M=T$`?.%:PCA%`8^!J0.^6/%*F9^]2)[,4%SN#1LP
MJ,*5PX3%EB'M8%V4%YC-,>0`SC1=;ML1.)M[<DS<TGPR;KA=WB9IZ$RYH0\(
M^?!@Y<?T-B2.1__$8S[EW/2OQ+;[\`UW*-[K+0C.00PZ45TWR!24*IGDL8L
M;E^_-D(=CM8]./(D:=#@F'//8VAZI&K^RO2UJ#T_!>4WAH[PPH2U_J#L!6*`
M,IHOIUP=\;04?2*K,]2X<^3KQ<EY`70?KUF#J9]LLY4I3ZTKB7+@YC(%3_J_
M(QK^FM-U>%$-1C5\74J"EW>\A1?O'7Y__^9^JRFHZVK>1:O,9'Q8RK7?K5IB
MO+KZ:,F6U>:DJFU#B29P(UYH*_!VZ'8UA#R'<^.FM?&R07API%9*\YP7BDML
M\?R)J-UQ/RP?W>_8^[7CA<.QFW>=0,)&\(PT-,0FYAMQ&TVK.UD7]P<-1]_4
MBK][MQ#=OK\PM<VC8'2@:SOD-S1PW9I%:&NTNN7[*`:5NP>ESEP5C?VE4/&I
M0\S4Q$#W]651[%?>[(7<.C*TT!\DB3)"6RA%!@]E8O#:$Z5B=M>$/'<<XCD3
MS72J_1<#,?R0VAM-1!A!38;>%9<DEIM;_*SA!$4\R$GA:OKK<2/[D6Z!^3N"
M0WRU&XG?=LR+<US[Y9\;).N]C)I,3Y5;YYGEE"4/@FKSR2Q'4CX9P=`@5E8/
M7CQI@77<-R9Q(0A#`7*MI3E(*<*-LU%JG?,=JQ4G95:A4!K2Y-@-IEXL=:\5
MWXAGHST"GA]29*5]1]YY*FB-]1M0=5C$NZZL%*?IW>-&7@:8`D%*_R1UACHJ
MFM@;.RL/.KP.`K5X#(:G+LZO`V>L1BKJ2Y;3];SU`I,C*K=0N7M8'5<LMWJL
M"TV!P5)M#MD>GV9S4$S5/YODQT>;1+^K\$A+N]+F892#Y6B_))0``CA*.\I[
MQ&'0\%L(I;]G'J-VL`.O/6VH6OQ;WWQ<P+.^P*[@+`V`%E!EU7NO;&9;M^4,
M$QM7=';"(&(O>W/W:/,&=>M;*?QN%HR-6:IP[6V,TU-XD$A51I)&^>Z.%%.?
MB(O\LMWA<O;Q[AI;%)QAB1(I>ZTZ&ZC\HNE`32:47H7$S&`S?L6&H&9_.EE@
M)U>9E\GBL??:8MD@N"EX2?(R^'Y'[5CKH2'UU_7L@=4%9F]VL4NE<=L'OP;[
M=).CZUO_RQ!9J\[>CI1SO0[]F]+-XF*CZ=D/7F:Z9D&=F`_3J]ZA2_4BTKU\
MJ\$OA_=0G$1-GT]K!2X_WM00`-)(=>2CT""NI#3D#B=5^+EJYW^.A@MR6M)I
MI.NDPA@(+$0HA@JB"(9!ANC[DDC,P6570%O9T)@<TA!HXRX[**"MB05'8C*3
MD)-.?.\V*']SVI-/1X=37LZIHZT]787M)GVV>75YH-]"F!.$[J\I[B,F=BF@
MN<RE^KMQ^<-7+P>C>?W5E*[1W)FT%VJ^*?D^CO95R8R&V?4%]86-R0Z.\E*_
MFRITBCIZO`0K:OYA'HQUW%]^6*D.7EUMNGB<_[D%ZSRJ#-SS6S7@GOP]UGZT
MY_#[IO:62V*X;'?O<?5Q=JF&D^R*Y?ZNS3<N-/BLW'AH+/3/0$-'``/%CJ(AQ
MM=;IWSWQQ`_(7^.F2UA@)QQ:!!8&N+2)X!DH6DX6]$S#Q//4^>`T,;%>6(M&
M4PH8?`:&@I^A,PKFK%@$7\ZN&CB_J5Z5%E_<JQK'Q<22D_D(T--B)&81X``$
M3.Q86*K1`!T)%N%=/4X%;@5NC)H5IH:83;)HHJ+1IN$5&4Z0\H61\ZWY8KJG
MO*G&77)*W&.E#WTVD]FHT(1%\N&XJW1]T%T=SM3L,G)Q!$$4;H.+'7!@?7'J
M>J2AWG;?]7!9O;;YZDNMSMZC_T/7[>+AH;28K\6J^>K-=8'S[Z$IW=C=!N^5
M%99FKM+=S2(]?#3K:KW5::_)2-)$ZD.?GB0V4E8-MN?/)'-#@\LY*P4X'(4H
MGPV\('\A_D)<RB>]U).LOM[S50$`#N]ER+UH*FQY$*YGKSB>.7?V*IO7#Z4E
M&$72@DQ6^8Y_:&H/0(?C<JASL/,1+;2$H^2?9(`Z'QERMMU5%>_(R-:/#5Y=
M!`2_3`FH7J"U%(L[4F)S+KB3T!#\HW^#S9G3,J]76)I>;CW=F21OXE&=_W#4
M_/![N/KQ=%1ZED>US:-6),1AW8*^5SN?E:/L_HKE(;A^^Z;U*C.3(]VF6$&`
M?6CW=^T(S2U[@R.30L6![)*\S^YN*^C=:13]=WCLRJ[2@6KM%/MW"SWW.CV.
MU9ES%WTW^2(,E\[R(HRT138**.[Q*#ZU67%AE;&>@Y>#SAW&L5%-G@X4'!Z^
MD'N@[VK/0X$=GY2RU-@<S#G`JLSI;NP^\%\!"?#,.0/TS?![N9ZH>.4WTWK;
MGFN30&=`T/E32O^L<0"*`E`&T>AD=`$"M/%I@-JOJ(``"CM\T5A<;J!7R/2B
M98?C.>/65D8"SA\9$T"0SMBU^-%EG'\%.W`GT,C?^=;&;]O>)9FF:WR\YKO*
MSM%HK5:'R,KN^>XNSUUEN6?5?XP;:W]-9PYG?L^,:]C*#\.'!EY*-S5>K+)4
M%SD0E!@)H1]6US8-=:E(PN7*:]`YU,HYXZN)+#+?R)]VW`>Z.IHLS,QD%!85
MV3C/9[3JE[I4+5XNSKBN.KJD)FNP$?G53F>[(W61#T5HWPAU:,5]C#M_.KQ
M?[TE?A::E(1("J,+(2"``E@SM8T=^G%XH9.=*']Q?XFE2H+XD-[A'Q,OACJ[
MD'JE,I3.#4W<;=49>JIZT]/6Q=JM]V.<1.[J[.($AO-*Z!!:;4*XG:*RJK5-
M:;FMQW5*D%=`P*IT;BR(C&IBU6!'.(U>4%!5X]OJY!VUTNI\YOQ-D,-,BD.2
M/><DXG=5`A<=%FUKT'VA2@AN@R,7%!4--XP\=N113YSUY:G6+Q>NY^7TV\?=
MW?(C.-<SF5CW\G/PFN8HHV<SUZ>?RM6CM;G0M^<[YB5XH,GJK**Y>3@ZY_H]
M-;3,<WYY*ONXLNKON%NA7\Y=KL;NB,K]&__ULYZ#=9+4SWX.8:$2*?332'B-
MK(QH+>(9H--<E8$`@4+R56(K-?]=G,H3Q5'?:K01XYO.M$GDJ>@I5_RR^^['
M!25*:/;<28E@GF%^1L=.*(S+RR8G]^VMK;4KG@H&#A8"KK;KIP/W<_>:G:>$
MW+Y?:VGOK-]H>$CXE3]=M%$XI6^_^ULGKT6^EED?"L!V]0L+JXT+&:E3[Q63
MNKKR5VAH]`STF]3RBC!)&%]\&AE^A?&\\>[6^?Y4K_XWDGOFMZ$`)4PQ;Y[=
M1@CM"73*?P5V&SUQ<F/<F)'#S(*>;T/R>=?Y-/Y:1^Y#=6=Y==G>II65(92S
M0P00+`U$Y+%_,2RTM=+TXO#<PW9\=OX6H9!&IV[6!R+FC)O?"AM>Z2PX6LR5
MK/VGH?$J(J*;7R?9JD1WU]XA)2MN,:E#]>;FP+NFJ@#*Z&1G:&CJ4FYCZ[G4
M]-6FX"P[IY272H2%D7%]O0`=S8GU(/9@7$.S\2SL^N3G?)9D8M^SNQ.&5._W
M8[\[Z\L:9,\KWMSN3>E4/E0 5CV@+Q1<?**L 5V^?LE-HIGGU*GB&QQ'0$#C

MV='Y_;N0CP)XR-(GK#N]3(A"Q*FWC^/SM/4'8W-C7WOZT&0C6[ULFNP+%?O.
M%#1B9AS`U)&T?BK_2*IL01E\BSX(N7(E\$HQS/MJ-8V.P3:E/U&E;&C;S/%`
M*PKWT<8&0=Y/)T&:[7VSLG0_^`>]<([P3J:@T`_\Y53)097,^;5GO*)ZK*FR
MLJ1%3S\DI-131]AV\H^">+#9ZF/SO'++HN[6$06[+!0QX2ZK$/XYVUX-7E\/
M2ONX(ZJ*#B4GAW\Q>/WC=D'J(I5>B.(?K,OOB^^]Z2V7I9O2*P<1EL?"<W<U
M,&MD3Z'%XVZ([VY1L@/"D,GD2RFLK&'KPX9W4X;#%S/*=S)]T451Z[RYW1'G
M]_?6R4;6-B;F1;KC2*2S>AV*\"`[!>]/L:WV`_L83=:(447Z%1';1TJ!-!IB
M`>Y*@7+U#8<?5'7\]:,L/&'82=YCRMRLK/K:.K18<M)E[S6'AJ,Z@Q2_X7Z5
M0YI/J9\U<0U[&)>^B^MLW7/]S8D6&US.I3+8&SM5%H[4;;-;G)TWBI$9<7+W
M6'L4$9*H6.:>U3A:[NA8E)0TWMZ^^!(33W.`-<G+^4RDAOXG!8UC4(?55&$Q
M=TR]\L1<V:>4Z-?V.P9JQ[PO"NU595O(2V*@,%V<>OY)7H2"KI;O2?F!3GWE
M9-T;M=,VC0J+D!`NH9':JLK.BI7&<[^SD\GX8'S3%@MC8[W"*)M2],O&A87&
MXZ-AA;=W!5S04'*2/W6E5):[6:#`DF;EI'6",R'^_8K6<O,TDK7XUA?]+F_8
M:RO::\5)E4K:EY8NSFAFBBC"N9D9U(996XUK,@HO-9<RJ22D5AJ^Z5>4N*I2
MU[YYPS\S=,Y.VA'LYYIK4/2KZ9R-<;U2L%#^>\OI49?F[I[.O"GWCN^>AA\?
M-`Q+'9(T(J?`J*[>VQ]'L;!=GI'S"^GB(\K*:B,%@]YUF;1U[3\NW-L=YTN%
MG[&^H4%G&]/-[%O$)LU[H(@;3W]V-N7^>V1H2;!>07![(CHD\'[YWI2?YJ+X
MO+MK<?,BM2A%;UN:HMQ1*<@U-X,:!2892W:I75H:J4UHJ<AZ@('+I$6%1F($
M<&,+)'.G7:?MIQWDR4JSTYPS8@%TXX7:N"3]W%1K^U)[S$(53#8*,I+2'<5/
MBC`2*`:9$>9[K,IXK/VR*71R-#T!P2D!W;WRW;F4C?-6]Z#`Y*'Z:9UI'B&1
MGF0-#L%S\;3XBI%GS^+D"2+H::.)*!9!Z.@<SZLK!S%;&J_]9H>4]?+?S_L7
M+OL/-Q6R"=:JUP]^SV?$B9Y(E`O_5*.,H_-UJ$'-P=5U#;7.26;%9,NXA:6P
MT/5"JG6E3KG&6+T'9*A*4&<2L-+5YG^S/'[*8<(GL,D]73E3K=]@;-#XY6OP
MN9#B`PQ455M;R5'V*J="`OF/'VKSJ63@Y;@.@H\'5%Q=R6I35<G%&1G(EIA2
MU=A8;TETDO7=>'(R44!#0ZZVUW8?URY8XL:8'<_OQT?(X,]8[(9*(4DW"^?!
M5,6CQ[;'\KQ<+B'=$TBLD`#$AR2CI\?<9EUQ_&AF9M1L&P?5G2.;AST55K
MW"#_QB(TN';8.:S97N=AVF_!\+;LLERZ4F=B$B8XV=3DR%C,B*@/&?\17J"J
M6):<@<#!<ZC5*#5,6-2DU,&Y=*6?4+G34T#3LV@0+,JQ\/=+?.=W=RZ_5N;@
MJKI\;QRWC=4%#9+3S7CIZ>GPG3"C>*"@;V'>P7YCNDP^1#^M*B]9H8.:]@T5
MW_IN:]1Q<(-%W)B!`05W[%]7O2"]:P;#:>H___`''4_9M8)IV&^D^+2E=Y6L
M<#&6"+Z]2L?``6MB27'[FW>E)K5*.]SKS4Z[&F0'MFPR_)Z4^TB@KEE5_>M)
M3Q/)U;W16928JM23?90H6[B1O5LT7I1:55-MI$&%3DZYS(XOOVE@K]_Z&AE)
MRP<M%ZIJ;;I]G`J:"5FY>.K(E7*Y06-M3(=NTF)R)G:O=+V07XXLV+2P2G4B
MTJQ,G!`O2X>G1>'FI!1-!U_NI:8493@4)?NOA^@89)X1L*5TZ+&56.DDO"IO
M80_DL<YP@0CMZ\^H*$_I+.IT[AN7Z5NG6.OK_1B3F4U<*FMHZ+(V"/YU[W.'
M8MO5V^7DF5<?NAI2*"IZ$5-']Z#\Y]P+,UVU*K]F38+N7"XX+9XL+4SN%%RR
M5VX87X5]WA4\<U#F8.;\D<FJT-*IMKY6E=_QBRPHFO#:_V>W=,1,2KE^I7'A
MEJ=(HW+\R\:F4;6I-Z'P`&?W`]W]4<Z&!@Z=8:XF,X]H0]9H:XMSQQ0ED3><
MDW<T=[\(XJ/E0G:$8(Q)7VGQV<?KFW5E[6E5592\=G?A\"$-CF_.Q>=%-X&Q
M+8R-6"3)K>8)N!CI;'&UR^8WV=MT!E#1/?%`C\J&CK+N@OR#';NK=DY8,UFL
M.E`.S3=`W034%D0V<^K"L6[T'N=D\L_L]5+U_Y2#4N*%B?>N]5,*ZI/L]Q)>
MIZ]FG36YU!V:9;:7[TQ/+]ADV^:<"8NTO4UX[>YPIEL@\5*%CF^D?IBAO+BR
M]`;MDRK!73Q1\R]#;=:!!*VJZJ-ROEZ]WP6..*%*2:4VS`91B9L%!((_TDQ_
MR&+*?I+]'A(B6VC3$:@;1#.N9]FF7C99U3&UWZ#3,2H(',PLJ@GNS=:?G9[D
M56'0W0RIYNL.OT$+:)V'#1&$,%(%,/*_^@..CV<4S$H&B_G+;:,^S156%J\-
MK>=[)UDFU'S1<;S'*H&ZA<D_%&_89"Z;NWN<OJDLI=<V9:(/S=BST.?!QOH&
MTK6UM5^C_Z51-CMU.G7Q<'MP6S4A,53=VL`Q4\Y>J7Z7RK#!QJPLF)I<9EQ)
MRV_LZ#S-B;$0&=4(P#09E#<'H:HW7EVWPFC((%T%JC^L("#>T)*,(`D#7]P[
MM4R1%>#&R2<@Z,!0_!R%IV&VO+`X:[#Z<QT?C.S[]F;Z_/:!^J-IV\/1U+GJ
ML(H.59N\NY%1/H%JZD';[JQ+]4Q3`\=TN:/IGI:&*QKG?JTAKC_:N,HQRNE
M>WS),E;IBLFTL+5/R'(]02'<A8S786N=U%D6WWJU&NL_Q&&=M9[/[$Z=!ER[
M!F$^A(A7)*`)S580&'M6OM/!3.;BXTNO\G_].J5R^8K</^>:IT71BLSW4"FE
M/R*C:*M__@SW5N>Q[-*U8_'6Z699<%A(=;KI7+]+=4I+N'T7H%U_4#$-:B#8
M$A!J(I0J&T9;"WH1/UDG2&R+%,6F:U6.8[^GI_W\#UY&!-`?5VI%G(/AB*;E
MO5!7C;L4#!6N.&(]YL-,>[F#F:]97^HSB.(=>Q#4+&A,@MJG:CK+(DU5=#E>
M79F.TT41K@E/),/H]!_VZ]K7FCV'PTYM^C;Z^M9>GW*Q<E:E6"H[4??)LGK:
MIXQ6(*/IB`@<5U#^+W$S]J6[]ILVG=IYP4J1LJNJ((Z*<H=R2=5HGR[H8WZP
MC2VC^(Y)DLN.;:'YM+]S8N3%IKA^-'E'"*O,H'Q[[N7J0_+LT;3>HY>>7DJ&
M<>9,Z<Z@A;6WW_)!\:%)AU&UL;_YGLVIR)`>G2KM`N/:<^L-+R^]#0/Y2KM&
M0DETR%99$M["1ZH;'?(&+PNS]W/U1OS$*Z>.Z*$0<,]WQO_.7M*5Q%/V<)J-
M44)A6"%/=Y^`M!XW$1PV!0&S_>(3?LGDJQP$1$#BXCIO_E.^N)@M+^#>U;?O
M*%U(K;)Q/C)?:?2=MT]O*?0SR\V4M?Q2F.#[-<GX-_)W=P.=$'T9E?%0%G%>
M"5CL#@.`R.<BW]K!_2@=6!],',!(-OPMK^M)A1IL:H<6]I?Z#2%!R8\^\O3H
ML7+OX@TV-S[X'[HT/A,8[PMNHR1K^E1N>)FJ\]QKUNS#0<.TV073^YTB,LDF
MWZ**,J7AWMZ;3T[PK9&X:HDM77)O^[NCNP^K?;>&QL5-.`7L8W05:;0E3'Z
M;=A$;1%N/9937@%RTVSI3CJ)ZO@WK3==-[QM[T]RFA)["VMFIC4=;;)IT7$Z
M]%C(I<'6T(A/$9+;=II(`B@OP<2"AT?WKN_TEO,+SSTN_S?NHQYJPTO##5W&
M>_[.3DW#7=5/L+G8>#`>-:PA6!_!&38;7@;1KG&0"0@@ON\&A;C+OP%G(5B!
M1`)F`Q1<B*;<6<W9\?6]BK9VL';0P!7FBT_2#65^>M'2939"ONGI38W.KTZ4
M"1*8?R8A*]]]W1E4>;/;NI-<:IP&E]D9F8'Q;BB#HB3G7T?C(14(T10^@O+>
M:K]S\?]YN/_?1@B[7ZBO>V'5>F+9B(,+?-0XN=!^?5D\R.GA7@.R^!*$R9Q>
MF&K@&&FP<-'U4)2(H7ZKFZ#*L&P.0^)J$+B(HU/-!,+(C))(]7D3'D\@N-%20
MED]T<T/MH[U<W]>%%]>[-K.NJ\%MI?K+!WJ5A48[1H[6J<6V!B9&EY<7YX6=
M12DIK1W7+0XOPOLN#-4^#[F3ACZ+J(#=E&KO0WZ!D!U$^F'.@==!%TF2B,**
ML&L'TY8%3$)<\TI79]?XAYWK=)L%_>F;A9OZ]3-*P4QL4D+P#W1,$`':LELL
MZP=\O""G4ACD^5H_X)0EDS4-V$"WY2`#,L9O/[^BM=8UM(I:5I]E\[X-CX>>
M1.E-ZXUJ3=;O-MCSZY;CCQ66,F>E&^<>]YD4/DSC2.66Z?SG4%1JL.KUN&R<
M2,B=XI"%7XOO$.]"^Z+@W3.@4MSS^*H(!J828%/41R_F='[[/K_Y@32NKZ#,
M_*8+[/&I/P-_AV@J[8KPYF:UDJN2U:.C\C8@%04WK:_37;3@KR_[(<)?O?CP
M^&T/.$L0%3`C"#HOOW]\?'&3;!?IWN$D!I*UUW,)"NQP\H`C9.\GSSIJQ$9C
MPRD2$B)"B>AHD>'S F4=I<K-.'A`3YU%W2.^"P@!NGE+3!&^I@`62-:2ZU`I

M9%U=4V77#>;<42MU'.^6WJG[4UQP4RQ%TBJ6;%/7&[;F>,8*>O_OWUJG8ERZ
M+$E*PW3$'*RJ:Q2439-(``><X>#*D#^?%9^I+X/4W$%>L_`-X>$)2\R)S@U2
M1_8MW3I5?NV5UUG(L*=@,W?8Z7$PL2]$/?P:U1C`;1K^?"_MJN9_ZCPT/+1T
M9NR88/&RJCP>Q;F[4%W>H/&&--_?C+^\7Z`S6H]47Z99&@-/@4-'D/7>GIF:
MP#P8;WK>S\#86.?SCU(=DII._)7TTF1G72C_#U/N"$%:'`QP1#Z*.%L#Q?@1
MB(-@-DIA/G+!0'$TT,<5965E6K$:814Z!CDY.09:"%WO^V]YT:D$E9+&E<%E
MC^4/E7K!TQ^F#0RF5QJ4_:;UIO'''(([I@W\=MR"KZ$4?2`'A#JOHS9(5X85
M8$1_&ZF'QBL(0Y2!)*W3B;J%_W?*07!P#C&$H+=Y<^9N;-,;9)]C:6>FE44
MFM?SZPX+!^>ALZJ+[OS::\/;@P;V,36W*S>8!QEGY+TU_(_OZ[H9?8$1)[.R
MU(:1E[VLZ*=T@Z+-YJBAPI29U[X54LKRRE$$5H*2)*$8+*C<\MEBQ*_H_$4
M?9D@E]SU<[1_(_=;WZCHFJ]55ZALGU2OZ[(-;ZK]L(_SZQ0(1R9>N\"N?YM:
M$"J#CXX546,9$B7YQ2N;L0?!2HN9S*X.QA#CR>GFI:TI*!3DY<][?KMSKL>#
M]'<=0YOC[.&D9R^O(4/7H^<UP2[E>_>"G5B00)0)4O#,26^5O;$ALK`0XRLF
M3)CBZ2H>7)S%[2>X:\H.MC1K.5"90,[WIF\OI!DN%XW#;M(;G^>N?KL)>]0W
M'E51-HE(G=YL*'R;5SS3^78MT")UK7USKM),>U%?Z6S"S%%A[Q75EH9`\QC'
ML=8C0NTM7&YJ`(U0@1":3G:R,PH0:F//WOL??],9J3B!4?'B:DZ6+0OQE\%Q
M"8A/L#Q!=+RB&CT/O,H$"&I0-1*WAI4/`Q,,J<B3!>$0#K]'(Z/(<6S_V14,
MP3^33NG`#^EXK`KT7WU$_C]02P,$%````@`G&/'&N5*/*M1=```8G0```X`
M$`!U;F1E<G=A=&5R+7!A;%58#`!+C7XWZ(@3+)P/9``UMP54G,WO-KRX>W%O
MD18O[E!**0[%?:%0W"G.+@[%W;W%;2GN[EYL<7=W6!;8]_G]S_?EG+E/)E<R
MF3GW)),@8>)B3F'D(BH%!00!^0"O:(9IA@???:06))\,:93AZ0=E:10=+A!S
MDBERYFNT]$W:VV]I^E?(XCS?<5W>_G3>_;#Q[L%`MDM?D?OAW=;;=ZY)6`FD
M1%U?-'`+JW^YG5AM6TF7?+&SZU<H6&W]@I0RL$A/PND8IA-!E*19<//I/'2X
M,#<@071F+[0T9!.G]FV^5`]HKB(-V[0,4Z:(T2NT%'-8WKWN.$VKE_#YLQ;J
M=*'/AWG%LA!&K\AF3*_1!OWJ,5<LC$"H@.Z7`(-R%$V\=$6*H(=?D0&]6F[%
M/GPI/*A2Q*1(^66=*@T.;&:8=,QLR!Z:G#G'A_^MYZ>DA:I7TBJ1$&%LIO4=
MTX2%#7F;=)4HY/C[?V;^)*1(N&5T"@WR_YGELK(AJVJ<)1WG_L],70L5LTB4
M?5[O?]OX4XIYJ"A@7'Z@@?7Y4ND;44#:\9"66#%QV&;![[>FO//E970Z_R':
MWX@V,X^WM4A*B,,VEK50F4I:#>X\V0H(4PB?R<'];,B[$>?PM[Q'W(5TW;*_
ML&4N+;S"UC]EH\]A;6Z^=C%0Y-R0=/?+$"8=Y2+A\0(8XY>8-?.?_*BGQJ)I
M0I`VS:?]*)`#\B!=4QL7"&YA3!0`'3(Z-CXQF0$=X#$:0,214$:]W7/PYS!)
M4!^@]VXS/H!Y&0F`-*I+H#*2H9XR$<=SIWB7[:%9K#--_$NZ+J0D/E;&I#0$
MO9BWYG6\-I5'E4N2,I`038%3A!2ONRP91?M;IW:E^I79`![IAG:[^I7Y`#XI
M0GM<_:HH+/F?DU,S(:_$=KA<\!7G_`*K.]9%P"`YJ9G*H.%"UW7MW"G5!!99
MHH"28O($,1G!&BQ>0"L_F$`ZBI`TPR6QA_B!1`F6D!WZ438PW1'U%R$IQ<&*
MOS5:U8IFS?0M>`R8A")+Y!]24X-1C\`E>L]3(J*FU_$11%(YT_J^DMRC23A1
M+^[[AMJT)#3=P%+VFN(VY'*\/Z<08#%-%#R66Y;S%=E.2D$R.)?OM\D+VN=\
MH$*:V@.C\A#J9<>(TW3?;6%)\C4CA)H>BZKK55QSI5,HF[O/N5.Q%HE)@9J>
M^D^#7-,E1%(?``X[SR/"Z6$`K&,$W:V,9<H<#[*">$V/'&HN68QJWE.O2_NY
MUW3R[RN]EOA0RVK9=%*PU@K3\FFVEI=N_=L+"9LNGEE!D*B=A],EL'K%$Y[H
M%M-$4%:#$:]D.?476+08])YN/P0M<X8Y=LE)NZ50WGH`O;YF"5MGS7`S5D3.
M).=I@P:!]:86AF`Q&!@9^'3:DMJMF/(HLO`='_/OT_:V(LX?$F\=++\<3:P'
M]I-4?B8X(M3J((-TJP.;_/+QC\X:%ISU>?EH/>^)F]VKU!OEQTX!L$O_KBX!
MT#$V`_]T+MDC1@K4T3(Z&\O4>WC*`X3EPETO3.9J1SWGSLI]71#E;?$SK7W5
M?K-4H]@OHT*:#+O&AW@4?[,VD7T"CGS>-A&;!W1$'`+17C,&U[$9#R]#OW=^
MV"LP[=HQ2"?B.&%_=80=XT18&7:**]'&^Z)/3(L.2"P'D4DQ4'&C^B1DC-8D
M)60RE$3;U/0:/DXV7D8ZIQ/J:.HG1UIFYDD3_W#W%C!0["[,2W@)/=C2ILII
M_/UK"Y2V1EH@?I218K%)?Q3E"A*<K$T_4OAYX'/1V2D$77IO0G0?UZ(?Q7CJ
M;R\3^A;(JVU3VU4?I9IS.J,(GG\0)'N48^LAQ.PF8-!X5?_:,O-/[&O:SHA<
M2RCEY\8.-$QF)A4E+Q4+6P*I=8[<3GTO8J&QI/60A@1ZCFKBTO<?VP.%.IN3
MHX,L$U6V;I7LY6D5DH?FR]9?U!'[;5=O/FB;&MTKT^-Q_A,;M"/7!8^=RY$1
MNW;#M1]5LV^YB_R'-4F)$SU9$UBC)?$Q]5`*U&5S&N[C^&<ET2`PPK0T4ARM
M2OTKZ8$$4@PM9?V/T@-1I.1:IQ2UGZR=2GY);-9:+FBP]^'R='P6QK$&?I7&
MVF=B0I%AQ[Q_EY1!,_#3X5TM?*>]!4B5Y#?1*`H4F@#*+F&+<(IJFE'A,E!]
M+$7K?;8.A9N2L0HK`=TB"UKG+O$Y55U2.PY(9'Z6;5(;9&US^$ZRAAY^ZR$#
MR"Q3;U<\]8@R#XNQO=_*>]=9?>"L&D)%A"MC[8C);2="0\(H%Q)(^(8\)"W%
MB"$#\R.#+`\-(1T9KS`+,NXO)>D70,4!*[[6#3L<)YL<X\T9N.&!M33>"!%Y
M1_!6YGO0[VWDS^9_=E!D0W:9?@PPA^W+#;&6OA]1.(Y4PIVHBG5R=DFLNQ5(
MHV@UI.U"H`!Z\%R7.::ICM.)>:0_H5*\XT>7BQ`50Y<\J1DF+OCMLQN:N21D
MA,6J3*_IIA&YHIM]?P[#(I1?ZN*.1(9_-<3`UQA.>-K[;H_F:H"W3:Z!O"4N
M4TSB)QKOJMX0@M&&0I_._$+.5C?KH)<H"$1",TPJM#@EB\\3&9)QH#@6?&D^
MWK3KC6EK1RX-Z1?H"S":VY+MPD&9"1,Z3%=X*C<4EZ?;(8ZW_6HA'3/.@Q,B
MB/3PP_*W7\C%8WI]?@#FL_6ZU],*=?W=]GDJ^'LP\-(V1C>H%^?("6^18B4?
MCDV[[Y0=1^4/PF@CW0/)W`:^07I?]?GW2S"F#'D:LA3*9]2G],KW[>^S+]/X
MG/F<M="X`(&,GQ[Y$#*1FT_/#*>W?9[]`>BH(?8LS^]QKZ!,BYC,>=Q2DC0G
M*ABDWZLO^]]>Y07X=;:+^+]M=R-DS#^@Q5U,X'-68-`.'=;M&:@)XI=&5IM@
M&_G%/6)HF\Y&W&@2B8JC47C9\OT-8P]C!J><"$4DRMR=920J!C(),KWREP%Y
MD<E@G)18%:)&<8H>^#-C?+#Q);D1OO%Z+>8_WK4%AKM+=8:9O"^1GY!?20'2
M4EUCT)T+.(9UW]3*`8[I"[<\G9H_9(-`F)!"%MPWZTX@F(=U34%)14V.(D;$
MP,/U%#79(8RGPB8F+BEWK!^)'(R"BH9DX1R?_N\?"8P8@$_TZ3"`%.?F)A]-
MMA_WSYMXYUUX"Z%0AYZ0,\2AQO_>+R5UOTFL-K++WI%1*(Q>7T#B.S\A7R
MA";T5;BE``F0_N=SB,3YE^?--&(F'EE9_T&BQ!?*C\;OZHYLL0#T*!5TZ,['
M9@'IZ.P!KIABC,1&*-0?`BI,L]&U@@0>*7CN>*0UXY#%2[;-TN_8MZQ[Y;`W
M12E&`(KYZ-#_\3+.3/)4E!2"&@#RH$(9)B-D^J!D.G3K(S-`.EUP'_Y6T;=G
M'-W^_RGR!'HP8>*QM[^\-B;T.N`D:#T@WQ,K\E+%EL9_VG'P=+`X68U=<NA[
M%VRQ<9I`4<1D8&!FPF#AI8;Y&05S$((FQ":4[WR%8":@-*7PLPM+"9!>4PZ
MH4<2?\$LI4$+W'ID\"+.4$6%LB([BV)6$]#2+T<D&[,LB9RDOXFVJ[^.P\7Z
MB,V-S8G-COT!FQ6;&?L=-B-VU^?PWC"9L-Y0F=`FA`@]-;WH[2<4$1%!G>^R
MOX4%@X>E!WX2)?VY\/958-9T>A5L2;YBT,LPP4A@7>R1FMJF>5LHAQQ)F!#B
M[O;/-T8,BV9D5XJ(8H`10&`V,A7":H:%LXG=]WE+6>6.*?H[4OMW:]OI9#,E
M"N)C%^GD@&2D+Z@![RQ+M&@!N1K,@)!E"WLL=LKL<B+DFLO`C+?>ANP'[.)N

M?;ADXB+.I76CTLJF-:\SU_.('I/<EF>!7A&)R9D/?3Q];$6-Y/>JY@<<,5QB
M6G9_8^F1P^CYPH'V"T[)*F\Z+$\?@TG7^BLD=+B6(;B&1`PU$A\Y'#\H@BXT
M$H&C\FE8NKN&)D_>.MEE7+FI)P?YO8$7II))&`H7LG9H^K&L-+Q7!C4']_Q-
MMX88<N*SLA>Q@5"^:X&">F0S3W&0:N^7@')>))44XIHS-72C$)``4DJD3PV8
MAV6HCF`'565)ZYB(!";#,MQEPVV6J@O!R.P@^:^"O_9%R<GZ7YC7;A=(Q,
MXH3RFH,?2[]QS<B^D]BE(Z5@LWJ6MLUV85">WI5A&MS-Z$<`8C<P\FL!U$BG
M.$$#IM\M=,\[_=\">#MZ%_(2D.!!J((8IU:&$EQ.;'W(N*/^M54X@1,C%W"I
MS<<@H5^3<;IS&]7TN9Z7M]FQD1S,>NHC8VF`47V5@_$=OW2+9RH_0$[`2_S9
MU4I#?3<][T]LQ<J32?$!;Y+U0X`M8,@9628W$6>Z\<KJ^TC7BL%<3@/,"K$!
M,B]AY&7<0?^%SUAZ4O87B6$RYNNNKH1_!_V"@A&Q]-@._N(Z<0TW)OG0\I"D
M4(:_UG+M5F:\H2O=03N'VHB8J9+YK#B^.[L#IYY^?`XL;JZC[=D1&JD:@6'O
M%523X$E'4$T*"7A)[8F!^1R[$L\44]300F)*\H#(]^_$2(HGLS=[@,/D,'NF
M"TF.JY\84Q.&W[>"AB3B-'F]T$X)DA4!#CZ7<>N+]&?EF(M32O?W>F^&[*PU
MKZV34!JCIU('DGCN^QGHEY%@&_(GT<0U&_NO(N^'MD6!-F"QJ.&I,V/4T[0C
M)QCJ-92X!DG@R6JJG67$*OOW8E)=^/JX$LPZ'FHDWOR9&$]W#8Q)*V;;&>D
MPA=);)-Z*?>9-6+'_*#)DYIX_?C'IE?>/WS52]@24,\E5RH>X[2-"_4^V.Y9
M?`CX,GQA<]9M^V4X[26;Y<1J)5KLK50VA\FHVH$[M5.>BALQ_P/$F^;/#TA9
MM6A@-_BS`G/`(37&G_K=U`=_[]`E4#3SUM"FP?[-\]4XN=-5[,USSSWU`<G$
MU:]G],74Y]%NB^WL`?Q%B+>16VJP'7+EIB4288A-;I_KO"&WGJ98F3_OA!U=
M^,8/I!S/[-76K,](!Q.Q45-01EV6;KG<5Y%9^A<TSY>M$GK-08V`B=JM(AM`
M$`-NW#1_@.4B<]+PKH,W_N*4&YVP-[XU-G=<WRXZZQ275]))ZM`>ZMLNYGQ6
MTW\SXGK./;128(S3KRLJ,T0$C[0.T[1JH'88^BFKWBI&"^=$Q:I45Q:O:1Y@
M_LTGQK@;S7$$1!N0J]1@=>_M173;%*"A#S`0*\\D8GL_$F$;_U[1X`,R!=/T
M9H(2>.C\A6I',J!;=OB,F6BG\P\20^(R*LK<,I3@+'#/KK-YZD1XO7HG2/)O
M71<[`O6<0>J>(A3:";9FZ[J_,\]'!*`D^`KD[A)DJ"/+NP0_FX+\2H2Y6&\&
M0,M0-8_S/;(N\`/2F_-=!BG<I#2TG_</#N10EVQ(Q.PKMV.&ZT`G&+C*VMQP
MCG,R="Z%*YXU.[T(Y>Y>:8D8B>GJ67AEA#2NOW*2BU=4!S^[^B/O?6S7%F^/
MC3K7M6<GON'ZA(V7==!Z-@ZMA^0E"M9#I.+G0(-K7@>BJA@P?0.2^YT)-)!;
M56S(R,Z"77`,WS'7UK/'/1Y<AO[>=ZS[9;,_Q:,3__M0QP?A`4`_<.%&$"@U
M"=5+%1\#2!^<A$N,[+B^7%IE(G"[^O`],](=:F^@%8#;?0MI7J`;P;GQ90^,
MRF';_<!N]@3CC9W1X$,2A+Y$#G%H!THD&(7&.5VLI.=W3+521TNTY%-9P176
M#^V`=GLGY&*QX\R@&)YSS4[P6"`&!";']:N(`G='\LUC9CL37?[,PWJM2VS.
MC[C8XKI=),3(B!7TN"V.Z-3+_X!JX(X"`/+\QF4JM63W@O[WE8!YH,N&"9\:
ML,OLZ6DI2;QRZ#[_HE-V^((OX\:M.1WB[7F.I.(`Q3??-7Z\[UOY3X%<!Y`Y
MLM/U(B[\7`A9B[WIR(`8<Y$-;JG7D^CW9`C#EZ8E3GHC(U(1QL@HT@;BZOL\
M-=-0H^$-L?0Y*+T%?..&F^,C0R0A8]=M=M?Z1NS8*T`)])+M^D)SP9?\43&6
MH%N4LXAP_G[*Z[P$@`G3?F2^/(APH5<OC^`48-X5*FP,^;`]4O'$@@5R@$VP
M8L%HDS;LMM#P<E>'S*G1WJIE:=4N/>]U'&'Q51%E*`]/S<,+3DXP!"/\UCO<
M'U%&YD#!+L!?)9`SK6D@4.J?R5"P$Y&WV:IGC=??'=R#F;TG[O,^?Y.YR7[J
MN[L[C`N8ZU(7T\A*4OO='=9E;21J0&+/`)>^2][LH-L`B`:AR.LF`_9ZV]3Y
MNJFKCR]YF`1;=:?QHZX"/F7>;:#\AK8^/26P#,O/Z0Z`S/PSUBH7SD.>GY\X
M8BY/A!,S[(+MP88:0!=*X*#D<'!E8&:;R(E")%2?[:B\F?XV&>HRIE1/8OQ<
M5FMZWKHX=M&AM<)\Y_#(=HZVH.)GY.ZF&6=8T1SG2>]:+X5Y\N%0Z!7[1=?'
M?Q(JCSX&-=+(8SOG\TQ*".GT-BWLX##EM7<CC&"X3@+.%:D!(N'5I*3:MDK
M:%'L`7G^*S?^RL(-,;YZBOT)G:G:9DC6$T:'=KJ[[U\Q2(XWU/@8"&O\2`-*
MM\'3D5[ES1>04L@/\O8M^DHOW@X5U9N+/]LF-9V=`*:N#MC%=1L1=PW@1H_M
MO((<4'N?@#2]^1IT=X=VTA[TVM#9EK.GZF_>I1\("NW</]3)-];;9&SY!#Y]
M.J5"F889N!_J$N^DLC_R`3E>NT[N@=:49QZOFS@?F(ZSOAL"NVX:V`EN[^ZH
M%9D,B4D?)K1&#>*IETS@;8*26Q(7?R'"-LAMQ8\[H,%>:K2I*80*RYV9YRL8
M].42S(@F`R9^(8G_Y?&#U(ER,4_Y<N%J9#B,\_R/S0A"<+1K`#1UI<'U;\&'
MDF#=<.7.#2A%`_$%`FMN&P=`QC7@]5V%#8-5:.=])W@`)CF>'MWMKKR1E7NP
MUPDN1&QC<YW64)I,GGL_9TK13/5M2W:J[I/H&3GC<@UPMV]Y?I<V@N0P3K@E
MG>I=DC2+LQ?F#8#T5B3W&[FCX#+#:J54MG+7JTLMD)<PA2ZK[^EF+GD6VL`X
MA0[3LDTP;'9'R-YPE?7"`,'.U9XC+`DAJ3T5-EPEV):80]P3*-NU0J;B#TXP
M:FZFZM>@+HW//LS.IT-,4`_RAX'8+$CG?DGS<L7VDFU:)]A>[K_S4%0G/C'T
MQ7[.L[J<GEH8`*5?[':DY`W%8LQ`Z>L+',C8D%ET<<@?6@R-5W%FH&IXU/_T
M=5>.3E@]:&IJ`TQWU@=EXL'_09L;`P,OFRO?_LU"X\YPD=35*0&6JN<!]L`=
MC/L7;Z/5H;G)5;X;I+5W"RZ+65T7YL*G*!6=8*SMA?:A-/^]NRGYZBC?'KZ;
M*WUREV"=;B.]1ZM/Y.V;.HH8G:2_,(HYG_@`4T]/00LNK1+@YVP\FM&V#Z8.
M72^&Z^3)T=9,,N#69H*3D_"B^B8(05XN>:K&ZOFVY(WLG_4KU9.'01!=ZBJL
M;_TUI])\S!V^QSIYC$QCM-JTR\ZDOBM'<E0G#\+U/FA><`F*Y[QSDECSI"3S
M)I)7!\.'51LMO@[UB4[N43GE/5C;679N1KU4,5,!ML\$N%IV_V%(_^?S1J`_
MHB.+N&,T[P;^6E?3WN\8=+=8M2T9S<M'SQX`%27CYZ/_#NZIVME#RN/S'LF+
MB5/@X=MX>V=,`/CB(E6C!';-\@E_F')XZHRP->?%"AP3T>L56>=,X%;S[!04
M!3GRZ`E2I6_\:W\^&.<:WR+T_K?PMVHEWA##*P;OB,L[=;)(FQ\=0+\KY^2
M6$7G=`0ZX(4\Y[YXR\LU&+V;D/;[/#MBZI<[-)GM^1AWY./B7=6H))P]MFJU
M;+G/=S@VCA-MO_5P-N17-?5Y&'*T[EXKL4%5(.H8"<=D](\/=J0UPH1%W,NI
MRKQ&&T8U^1_8"4R>D<C%;1SK\?P^GU?[^+J7,E#%^+)QT\"%V/$3D!P`;:V0
M7R/F"317PYL)60)11G6F)DO"43!/P>%/Y5+N4M$)%S\/IMX\H-2X>\<#R3%=
MJY8]DS.<!-\\(*FB^I.E=7Y[*+AE``8T;JW\N5&0H\6XR,!GHML]H/%\S<QP
M&HWCX^-6KUZHGTAANOL6]E]:>\O]`RAULP0EL-#9$\XD%V^$+^O@)U7$WL"(
MTEII!(-M^S^*;W=!@->@E\C2*5Y$-#4SP_`FS=%-@F16UZ$ZU&4;2B`.K*'#
MUYD<'R)[".I\B?HO)B+@Z<-Z[^J;U=RC_JL6Z2]^\;;=A#=>7;;$GK`MQB:P
MYZ"`22\\ZTP/2+6N'#H&0,I&P^XTST-[N_`1Z+B#'\OJ:6BTSL#&\+6`GE>Z
MC8M4NH,D^P"TU[]E.&1V6S(?NBW)@?@%>CF3.CQL\J]D!?[Y9O3\Y*`:X7L7
M-S5&S"CU1Z[?BB/F6J`6%/+\2-5EB#$I\%]HQQO=KE@I[5Q8\+W=+?&C#8FG
M]X</-1SLA=/"J("&+BV:E@\[K_T0[GS[#>E2?OO!`C]$AE<OEUXA\Z<PY/5V
M`JAXB\0>U38T#OONY"O<77.)_(R8I[8*UC[-8?4&.]F/T][H']70=MBW^\N3
MKREQAD,5%<HLRGG]A/XFWJJSO]&%>KG`(*21YVY)*?C.;3M*2G,K?9O^JJZZ
MG[EJ+6)*WEU_E2!V*7&^X4'2M]JM-EJR82IFK*!^)/.B"5$8BW:V!86<KOXH
MB#7T)X]8X199RH:W7)J;.B+7@' SO)[.WCZ[J`X3C!JL"AF>:/GA3?!UW3M]

M$(G_%Z`TA=P^`"J$*-!:P;@XS@W-#Q`[;7<\3;0AZGO0B\4U4<1?%Z2N.BR(
MVNK7DKF=KW+OYER$$7^REELW7RH<W%TXSE'4UV%VON"7\L47C[FNBKB.4O%U
M(^45Q9J0P;0QOU>+1&#7CRYLJ>>Y1ZJ:ED_#NK=)%W;?9_S=B![RX'GE!I4
M[\R8N>5O@1X*B_<!Z/?W6.FNK<EC1[SKA'"VWTV1.,FUC;+Z]N2'W<NE`,Z
M&^"O1_NFR((G&D,PXTY/8KF-PXH=>Q=Z.['KR?Q/5`I1\'+6W8?9D.>`H2]^
MX+0M-)J@N6TPCW%24>)<DVBL%1%&I`R8\L@SYZ)K_>KQ;ZAI0FA_635%JJ^
MYRC([:ON$%I=S-Q!I&_/(]M[9U]]W=4TT4D-=OJ'!#<4XT8,11GPQ0ET?P-Z
MD?#DY3@`HBN[.!$C4&/9MDNPLSP@CP[3W/>$$YRBW*W#GQA+'UQ=Z)%(AD`O
MS@NLDCX^4U/TP+9MD^GUUX=U*/?C8P#?_3KIPY8?F6^66X3+Z1Z4OON3//AE
MNM)#Y`]$P.S7K"MCN$>5P$9&W!6M8837([X2?[W64&`J.=E[<O5%OX#R'=N$
M?G_P)&RD:6KGCN"GW]4)AFSC+-1O)K0S[/'N7079C.XFE!N$1]GA/#H`D7H$
M5NR<>U^TU43!"R%XJL"3[4[>[8I"*66W^M<ZJ@6<(9S0<C%?57J*J5+H13C
M?.R:Z_P/Z,7D^31J7RFVQO'@HH5(4(EM%Z-T`_JZJ&^]I3/$6/[8#C([_[EZ
M0?'8W4]`26$]GE<]:]4%2ZUX!X0_!1N>*7UX9=OU%//W2OA7T@4><CF.TI/J
M$*)1.S1>);!FZ]P&:W&D+0ZII!NO#G6*9&[;O=NO$26KE!I;[@3W87U<8:0W
M_70'_Q9_(9_:']X.>OE41.`DE(=R=4<"[)8RLGO8FGW4S_0_1&69/LHJCM]H
M+\%Z,;)=HO-J,DO+@OP3"%K:/A`H8231HKG_FKO7._7AO:O#RNS/RTY^`Y/.
MG<"GT&5TF1X/R[58'%@@GM()1I,3<?-9%70V8+\8/*9*YRGU\3S8H\(.XJ7H
MW!$NJP!JC[AN0$7@G:BYTF==P7F`!4_,%)RTPG,Z+5TDX[.`NZI$GT_P)Z*3
MKB/)F(12M1O6(XB1U4'2QYN.^`V]B3LE6HP\+[V$Y4O_(/OY2=&XRJE.FS(<
MV%6)B7<=W41I]/U6GT.;XX9+CI[RQC$]!9)/(\N&=]QO[SRF09!SE(N4'O?6
MYT6K>('2GK!J8&T@#/P6XHG`VJAZ1AUJ)A>>#.KJ2]-0JLI_G/D51Z&W:Y=K
MF/D#RV>8HE.4S+YMEI,:-S*_-+9OR3?'"@[G=T>4/6U`G7`4>Z\%*K-P7CU!
MTDP*H(J=<K=6,U^@5/HZVAT=)'_MX<Z0]FLRAJ#:-R*>\QL^[NN&H.V;SLY\
MS57-9G.*!^3R5C6*!V^@E-%`_\0RTD[GR[;=0)?5/H;N<-REP"FWV>K71ZJA
M=5;P2LG%M\,^:*OWU'MY?)R?#W=]33M;]91:Z"2-WB%5$RKD(AR3&]`I;Q,9
MFXN3K[?]M^,+(2Y`QQF6XBWULAA->HT/ZV[HG7EI3B^&71#NP]+#+3M(5]J>
M=\/0QL!_RZI<)OMGZ*#31:_[%T^2=V*?-ELI'*]<D%,*(2]>5PF;-SQ?)?G
M0?QPIYGT40%$3.8;N.S2)!]7="`BR2C5_JR),!D;)UB&D@/(L/TF8J2NHK"P
M`O:53IADAPF*<^]#7;B=8[ZLT4DW!C_O?S]GS>H8V3A1H-5IR=/DYMZ1!(T/
MY3TXT0^;8[T!)L9A6+XL7H]Q@>P(-^HN/%.S=,+B2ANH!'=KGITDI<1VD%,%
MCY8F/*(%&U`&AO3PNMK'RR0GF\D?P[YWM?CO@AXH)>\XPZ^F^M??_8%1WE+J
M7)_@A%!G3`4]3C:K)>8]>*;;;48L$H.=$N[M`_W"T+S.?,767?P2_UGW@+U"
MFO^`_*=\&Y']A([/1/=)#6;)GVS>/7LA6A/MIK(,!1TW=P6LU@N>7B.2GN9Z
M/1RN8OACX..]\JGPZ[2F*'8$Z.599M3W%/&(9E2^=L]\L[P==$+L'"Z(%3FE
M1J3N%_1B=RF`D2<!:@]=$79HDC5^2Y8JY3F:YA<]OFK3#)'26^_)M>F:@HBY
M+=^,6"BN6JZ#7H1M"XKSPWV?QSH.5C_4'".D71(5.0DSDU]+1^"%N^K<6TT
MVZQ"^870L^!/*%F=ZU$MS&K'G]62X$\ID`R@Y89S,*IJ""WA7^2\YD>Y-CAU
MQB!H/G==S%\B;[Q!J).;%Y?K+^76!I1<%H8%VT/NCDUZ>O92C3U:.*3Q([F>
M)05[^R7"0:">NZPCTYJZ#]L>ZA=58^MJ;ZYWRA\'`]&6?"_UW^Z8,!R]N6;?
MD5SIS[G*\IT>:YZ"/R%=W%\^-+J-CPK8>Y_<%3`%X:[<O03F.5>-2QET4:^8
MMS'QG5,+<@'%+]PUN!F\!T$3^N>\O/D!7GFJ])U&9UMY1%0/WMPBOAC57]78
MH2C4XVD(]CO&8(7>/_LMV;3>%Y:PQH>+G),!B4R8\(ZDGQ-H3[RZZX3RP0QG
M5S7B[U[OZHN[U<<=D_P.RH<7[A)#I+0T5Y(OFH_81@2.UG^<&+OV!+Q]<6`2
M9K]15APK)6M'&,YJ2\G\0K8U!T$U()`-D/=?VOZ^OVOO3Z;_CC<;K@!F,3$-
M,&.4ZAQKM'OKF<C2?1;,>ZZYZ;/9\%LR^-*1]^J`C^G,VI![,\TE+X6YM+>9
MP*%I95VT.,3G8OYN_`5&@H01(8LNEH?8-.--"W;Q#ZB,:IHE?_2%C]6CW/_7
MS#\XP)LUF?W_?;\@&@2E4R4"1P_:(+!';@[>K&[Q]YW[*])R&D^V=W3%!'GK
M1;5+2MYY%I<EFF")U$6,<1OK`+NIJA5S:=?T3B0HVK_Q\,/]SNT5?7KZ^A/M
M*S9AUN;JY?FGINHUDO:-_1/31>#C&9?#MT;AJ:&VCA,5ZW$N3=WV]Y)U<:J
M3K[;K*]YJ`Q?DVC*A=E,C*0A'ITVPOXV'9/:7;G;1._R@-AW'UZ9D*'O>NZZ
MH2/_U>X9AMO<-PK1\.A?!?9V;8WW.6MAZ*-+-@G\>8/KP0X<WJW]R4Z20@[?
M5Q^,HN&IUD=^!7+#_]`$+0S=H5'8DOP8P%'LUVOJ6>R?XX@^E?[F.)'X?,DN
M]0%^59[/G8[(?_8J'"3">JHQFI,W=).O>A\&FPFD-XGAE_W-$`R#QE:AP9=N
MSVL)DD@G_Q\5'NM.-`M^W9:2T2YY_;I64R8N9T'E4%_5)3[*IM9GD#A:$'#>
M1.WB!1#E;<_UUC62!U02[_E1>-/'(UN4SV"`B82GY=X4[/>0LV\6AY:\.>8P
MT<ZPH#G,W6-6IPF,^ZP:B67R_3X@>Z\3)"[9/KNN0],#6I^;]>MJG(DEW^E,
MW&?WZ-CDO_E!2G_OLIE0L^[*.%>\6D8`';>R,Y]\FSOG:;_5:;//M^FCAUT%
MBO=-I/-D6<9>P1@(5SO61"7"AJ'Y*S*]N=-Q($CGK*I3YP.:#DMV$/2^JP3>
ME'-GV6@#*SD?JW3?[Y;LE[QP=U\U67KYIPFJC>D,&.QAA3Y+9[+]%Y,ZK0EO
M:Y6*;=88QK#N@J7^F#VZ5<ZN+<RH3'+W4-TH$L=@TU($?%B`]TNZ`]CYIY:%
MR14"3-$^*_4AR5+$FRI'8F5CL7X0(F?&EB8-C`@OX:2%V37J.WG,H^:/=Y)B
MP9#+)WGJF8I"`$C6.4E29@[>3]*K,\-3XQK.VZBQCRG8TDOU?"[KW,V1ZCH>
M!GS%\?2B_3`\\<95'L8!6#GS.ST;.49=)]*?G+/V'K;E['=R=^!SRJF4YQ=2
M#VT=9AU_',PKW`O$#QGP?9V8X1<\7U]C=M+N1\VCE)*2A>HR3NLZMX&:QG:.
M-/LV9WZIVZ]6-#][6KL[3]C</S("IQ$?RO291U=JXO^T?A`8VESF93.'J?[E
MKE1;CRBP`-`%(0X4*/*CA9XI`5ZD)HB36XKIXLV_,I4!2P'YQ96`5`%A:3(B
M:5E<0D:4"#06YDR#?>1AT973B/B-5%*1@]U?\?0TBTF?@F+B&[N[V[!G*PQ-
M<+$+&2N_O2_]1[;62Z*B&@]\[8;S"7@WOLNF_ZJ74<Z`/B5X\W-"`!AM6)23
M5O5@?(0X2/+9\R=TGV@XTV8,1T=C:']KIG36&V&GX2_>4&8+>E;BAX-Y!U6
MOYZ[P,_DTYV</@T`:3UN3A`^T/K&N>NSCAGT3)-ZO[:<FEO%Y3OH"[,"I*M<
MS$^(?5=WW5N0]-K\L,;Y1PFRI38CD`<XJ:AR<6/^.3_NSY2F'09[J,X7=+F(
M-^^0B8/(&$=P$CF^26/60,,I0Y=@UHMOT\].SPRM0:9FSD=78M8<'43FGYOF
M5]:"MN\6D#V.`@L8\;3>OBG]5TC&)_U+X=<++\^(Y;3%W<I+4'PRB[(@YOM?
M:CQTBPX;V^-B`$:7<2<E',5AI^!G#Z.J?W93SS.+MQ(7,7>=;G6JAPP[A5'6
MDUF`^(T%9,>#@[UBW.E"C:DI!SR#;]1QZZ_B\!(`T37<EQS*=]FY->QJT!NG
MC\/-_GB":.MRREYZ&JJ]?]56<6_NLYW]V20(;;V?_-2<=Z+]H="GN]QR^[%G
M2UO676Z<\E52@SS@?*4B83[TF5"98'^B?MZ>GRHQ=1)2S<T)[<Y_L*K?P]V\
M()O1U#-\.#N*22M!#T4I,"B\6#C(4^.M268E#Y!U,0C_1?7.G7\3[/;0Y\E'
MKH(INYDL:7!=@(SR/A[]`\B>*L<IH:H`3ZB?7(@9(..@8).4F\G=FXT*T^X5
MF76`V(QL\#.(V*"!Z^[YYF7[#-TPX:VH)U'@8`\7@97BOK!G\3]/<S11C96R

M2OH&[('2LT(-"(E9H';3?,,8TLS\C/4PE\H:Y0ZX(;7FP99$V^O\0'MAWB/F
M_"5L6:N<W7JL_+3B,UC;1D?K^]F:ULWM3\Z)T:8NM-ZS'X,*)]U/U<??HNE#
MGP[()YC;`ISO:24$-@M2Y\O2WF/T2'_XAD,BCI9UY?M1GV<X=IR7'#4K-VDE
M:0:'D=G!!Q7U4X^Z)",GFL!TJ0_);X^8Z+(,W1\.[@>*T_')Z2NY(*1"C$??
MF.&@W[^A7H27];EPKO8`,BQV%=E3>S<<T@'D8%SZY_8U-Q5F&N^\N$\IHKD[
M4"3_$NBD%;SS+F7M9@RA'S,[]PI?+GZ.GJ$/*$K))?$!+LSK$I`(W?FNL[^<
M!">ZL-3G93I4Y0#"35I"=7KFD&_;$+EIW%=?.6.^_2ZSL=R+<$FWCVJ`5%V_
MN67;7''8^E"7*MA%J6.,6'DJ9BMM"X5>A\]?W#:V:[@;H[GRW5?-JJS5WKF^
MK^3\OM:9=%9.+C@'2KJP^IC]>JDY=SS`1OI=(27D$Q92,'+?_<=_A0']3[I+
M1P6_U=67#%(E"3?FE(.1/_Q#)PP-LFR&/@WULI%[28^+(9Y/Z(OV4@-\/&V^
MJA`'92S&)N.Z\3Z3"Z9:R//T,N3+RV^!VYOD#\/2`DN4U+Z^71S37&<096DN
M*%Y6"63A;).51L,C\EWY\AS\4'"PWQ!,!$T)O;H?*$*O9DR()5;>FSNGK?O^
M&:#CD3)O!DC87SG!2@=:$9MTV*A4OW;7G_SMJ)KID(Q%UKI%I_`$V;[.Q\Q%
M.2TYZ2\MF.<V5+?B_.GJ*+;6TOWB=F?LON'_</'CAZ/W!X`6@1_>JL`:^`3D
M1*]*9>*1?0CZI9O4GCI=9OSUS"IKPNK<<(N>@C[;FB37#"DPVOK3]SJI=E.@
MGK6Y^JSS]9:)E>.:I>YZJ$;`QYHOIHJJ@S76.JG)'3:HW]D4E:AM47H=2V#K
MC,18C(T,(;+,8_6\?*=$W7[.HU\M7CJ[7S+LB7)SDHS=;K!ZR.E_7W0*J6+!
M>9FU[$AF^M4,%2!R6S(UXYUSD.=0VV6O)4-1CGL<6?:GQPV(6+>:F,3_0S2
MZ/KZ0%>H9UY%\($3<J-O!#C;N[&Q>!$MM^WVLS;U?IL:=@O3]1\^`A9V8IB+
ME'_0LKM/BDE55[`WM<\&BXV.`_!%MVLZNLN4'R5<-[*XB!6=`_;DUP#\\GRG
M!SPT>-NU:6.M]L8#?LQ3>OL*_3SWC5QX4\GI,+<\/JKU7%147)"G],(2P%]E
MQ";.22I396URXC.PSMTP?YQV%1N`F@0$[+Y%<[NAR0<Z00/6C#^6%.\AI64F
MH:R\TT3G8;#%U`"XA6V[7_D[$8/O7.["?F,=ZME%:>XXMRE#6&XL%[BC'NO8
M2+Q:)7U?;)K7$#C-Y7>"Y)=BN9TK:.(;%YU5^D\:7;&:]PU%#:MB6J)=5"W"
M1J8*)%Q<^SDCGF=6)A.?!_3>L\S"/;JT::ZI7R5I9PYPS=.64EM.3@ZYC_!O
MVIXDD]9U'UNR_W0>QU<$?]Z())@1*@_K@Q7H'G\XJ?Q2V@^1:9;"\MYD1IU8
M\D=%_0!X:ORVEG/FX>Z'O6!3T[_^UK#*2:.67P70'$8I7%D%O6:W(3JF2V1#
MY*E)WB\)ZY!0SV+5^2R-[=TZ6/4"W%#IWC!B815$>O\LU`N;4Q^45YM$PTAG
MN7<]5%U[T/>BD'8!5=@EJ0I];JDVZ?(I1GC'..[F[<U7=KDO$+(%;<*<[M)N
MNC;.WMLR8@H`V+XW2!DVQ#4@U2%_/OZ+I*!ICJ%R0NV!_DSYZUG4*G,]QI4`
M9D:MLC/BO[3Q]#C.`&YBCNR3?:X2@MF^D=[P'CXVI_YID(HY#=H]&9\TN_-]
MT!68[(M3;^`D?W_T86&>+-2T_3L&4DJ-R[:9F6D5XZ7;/$+4H8[1,1-J"=+^
M.[#A?I=6CJ+^8DZWJ>"5A"7*%O$'#[L]WYB*/`!@GO0G_L`V"UF"LZ&IHN%,
M&;#^2D;X87I$WC':2I,8"RUBA.B[<\T5(^#"\,BB,"[FE![]3KZ`18#?:4I\
MEW5$(;B*-E"3-C+P@UI-C4N/Q8O:297M\^N\@:?_T5W*4G$:<OAS(2E9W`]]
M0^.WH_YVK&>S2JSJ86(.0F8&&PHB=%7S7U]4GJUHG1=XA?1RN3>'\Y[Y=J$M
M$L_;XW&_O!)O+O\H*QFU[5:(G5:#\&_8`9>]"\4UL]Q(#0;U-V/PP*#!X*@(
MB969<509O*@W3:U_9S2;""AR%S6VT[A#[N>?S,LS%3*/EL3R/0,WJ)HT[<)/
M[Z7V\H]LEL["NA$->!UM'5I<:<OB1Y]0%'%XB&DLXN6QJ;[G1],*^AW>EJ[<
M),ST?OH<B4$?$+G!*'5[/A`/0IR9U)?$YZGZH@+TD`,JYB<Q]-I,3X)2.$FM
M*NW&529HIN<F\>J(Q9-QE1SV3EREO>Z:/-C'\T'6_ILUG<;=?@T:HFL/WR
M@+25[=L39]"VKGK.HXV>(0/TT>\T7HIP4A7O7K@)GHT#"'W49;Q>(=85B89E
MID\M7G]8ZC0UX563>B?%3==F5R!!])&YWKE9KZ4D+8-JE<>MAC@8&0FU"R+`
M;G>EHK1?@O.VJXZMJ>WFY!A).7;B@>Q'9DKS-[NQ]8_F4"R=1K8,K&]L[6\(
MW#%QT-X&FOW%X=F47HS#^>8]Z_[7Z_*_JW!9,V=7=%#K9.>@4`MPS"Q>JHDW
M550#$**CI,LSF-:J)36R-RC9G49G1->C!Z'Y6>"-?A7>\-,I*U)8,&/00I-+
M=,I$.?#.V1@IZ31U3$5?JW4#0=&7E-Y!.\;J;HPON'_PA`_'&4^21HNW7.OP
M;.JF71=>C]S/.W`ZOOZ=@MU,:&P,9YD:'D+HT[I`/X6VC$+,PY^3H?M64>D.
M&9%?L`=\?VA[B[P77R=$%^:<]0H8#T%&1J/18T"^V]EC[0VN^]?5VX:H>*UV
M+5BLGYLT$M\K27AM-,_-K<WR6OH0T+('*S+X;#\AT-!)/PTQA0S37M\<^=AR
MU5;_UM?)K)@L`K;YP\X%NN_<_*N$JM9TG'AQA_NF$@-O/PU,,8]+3]+T.,V0
M#)\>X3"B$96Z4&@3:3&L!C34B8]U5&[,!"[.^_%S:WB]GB`N$BXS+7BWI:83
M[?,<)9@55<[D_P"R06N[!W/EXDV^TQN0[CH(-[S.LW&4U=<(!Q;QA&!\?;I@
ME++KU%3I'IMVN#K*/*]P;`8%O7R!_7P=7:[PI37SQ]K%^4TF8\Z#*OU#G9U$
MHV#S_OYRJXYEIFIC_?4I04?(I>UX?,R!WIL[#C5SBOL^GMY;TB.CR,:C96T^
MTU18N#)W7<G+SBDK"\KJ-<RMS,\G],_'SFFC7OYN;P`CWS2H<!V\D),1<2OV
MD\/NIZ3(AJB6=$RU8]^/#ZCZ;:4B1W]W0M2M24V91*9BD*>K>?(-]J_<$F^%
M[A1-<Q8:H7GN>SCG]2-P-$>Q9RMM@Y(TI1064JPR%NG/X")K0$%R(8>&=[@O
MHU1,],9JR$'7*^+E6T11@IB=5)DM,KC5/1;^X"_*Q'&IWM$'G24>N3*CS\CD
M4F_,M\R.]FD04=)^'A=A)%D$O"<R-:V$&E85&7\PM82#+KPSTO#!_@/)]8K$
M;]$^Z5C24<%,/8LT=H?V/@W\>M9WG.C2AG.2=MMEYJ((QE659*%D%I<M1#`_
M:=M:Q5#)@7-S&Y$-2FU5FXOO'_1%NG%G.?BJ=+B[;-U=K)$N`L(^!6&%J'<O
M,8S/O>'I?3M=,UV6EQB`[`^H%"+[6?$B?B(F3L7I`%`TV$_>36-M3\!'[W\X
MF8??_5EX@0*MW"%;1!E6*@-JZ\LF*4&T*@I$\=T`O1A<_MLYC.L$JU09;%AA
MU_)G`M7=!]QJ%*.M^P+\5<F\X0L;9V?>VFMV=CCXJ2;ROKOEZ6`(D:1MN51N
MF,8:8.@;[OB=J7NK&Y/2U'G-V38F\Y-1Q;?K;WOW&NAPOD"L-W?(G0]-KU6=
M/[#M%QH.^U/0WGZ32<2N*\$6F7BT!:$J8J&/#,JDTQ,:<LVT<O][O)LR,!I\
MJ%).NNIP&>Q@:X!J8)Q5ILVR_6VXA]SUE,B%ZLV2_=,OC]!_#'M.2:-EU",F
MTQ?7F`2\?_-KRT+XPRK#5]9T"K*\];PCP"GY9QD%($9-X`*I%/1]G-R7B/.6
M$:O@JN?%PXN)%+:B;YC$O+5G)C$ZLCBER/RSXTHWLW`[J=OG]<Z.!T?CD_>`
MTHNXXW$'71\UI7V_[9JQ9.W/$(.$O':RA;1]MHE]G.%.FT?S5-S@'Q^NZ[9-
M2M*\N^PK"_W6E&7GNK4(T7F"L=#02=^@G-^=B@EL1P0CU3"8,\H;6+_1$L3<
M/+5RJ@*3A_92_/HA>!`&62"I]$S!OMB0O2BJ*H^O[[,E?UM"VGR$.,OY4TQ>
MOG1@6!P:ZHBU:N?8<A?=4)SY[E)'X&;F]>L=/95.->_A!K?`Q-?+&%SZ4B1#
MAY;ORPHBG+1@S.3A_QY(J)W_K;][HX1,0MZB(4.QT7DIXZZ>1RXG#2XU,[*'
MS]L=;1S\UT;^;YM>;:TLFZR%TT:5,C(5`<P0+<X8OIU/KX@AN96FJ=?[BRI?
M>Y"N,><:L'YHIGO?IB3$YPQQRXUZ]5*B8U?L7:6B^*,H+\4!Y:>?OF7BO9$_
M`_B]:D*>]$4*"S_)'@KE*9S86,UN[Z.D:TO$S<CM>O\Y-2/>U-ZS"DJ:GSP'
M,PMEIE5R>GRY(B`Q)AJ)AZ]#5@6R+OAOF*@78N-4\8/,V#&22,.A-D"C0=ES
M_;92R\VPHDJ7PK?1G_/9GG1S<;DG.Z$;>C'UF\/^_6^V,/LEWEI"AX<D8CV3
MB.?69;EUIV92[\%1R8Z3"NL-JZC"X:GW,%VK9<!$XX)BP=A4&-[-V;RO`#JR

MZ!T9;9"*@G7(&X`6+R$?C%3@I?9EGU7*NVZ.G?;RI\JR,<(_I_ILX^#`4(<U
MGTYB94!B2?)#UWEPPX=KL8[*"3+R]9W(9^+CM0/TIUDA?JC3!I?CRLU4.GYD
MU<I3X8!S6_D7<%'9WAWL3X&<:0V@!9_N/04Z6N^Q+.J^2,3(0C'E[M<O`)3%
M&&P783[US:V>(;6\GGR.1Y^C[R\]2DVSR?77SXK^+[9H4<_X[0W@ZA*``[6E
M04Z?MJ)&\&S('&,[=OS?4H]THZ`/I-(1ID&7N;\4!W\A96`"I0,K&YBU-CS:
M?*AWK<[]'Z1;V<"KZ'@W#:=#'<T?J3:?"LXVYCV:+LY\V#\<X`"RD`.+L,$A
M13/KMQ75;1==8*1*2"V]4!49[&6_;07;!"E[)05[FGV@6+K=(C16].?WYV=#
M8(OQ$YAD^MZ2N1OR];FQ7_AZ*;%34<5;*&_0#DU1E>\`'+J9_&3%6R'9`8QF
MM/?OZ^E+=>A\P.YY"CRVFQK%2+/+4.I)&\46CVJ\6:M`9S0@B$.3PT`186"[
M]EW<7!(].ZVJ4"#W.O`#%3T!=)6@!G<^7V_2\I2\]&'/V#_;0$I`:KW#G7E#
M["(T5OQY(4)W!_OXZ46F7;2^_"TA$9,O8`]C!\,@*%O37*A760YSA4[#+PP\
ML1VEC>E1@]YK+8=:T3K[&XFV\V14)D;Y"25D0B&J7=`C1S:-$OFY_<C;#WK+
M_5+LV?T\Q8=Y9L*6TL($OD%ZF,J%2'C1U&%'](#"?;\%1P@+;"%5?R)NOV1U
MWB!OR2GC19XFM.KM!E7QOE%=S,L"5VE\QHV#%,1@PP=C/>9F0O?OX$_.%O0/
M>W^D;I,'2=XX/\9B,=6^E=)F9K^P7F.U0`WE>2Q=71<6U.JU,$?!,;LGLTR-
M6KK_L-G8WR7A^U2/,XFO;)BT0F4$7=TP6/,SQAA_(E],>T;`LR<,VCH^GO8_
M`G3OY#FV@%X.FO7&]*<I,$Y0:DKH4)>NR/S3+@.XW'S^C$D8`E2%3UX8<7,B
MS'Z>F&`5%Y&]@7T*^X5OM!>Q?@H(9$3Z]`]+X;?<WJ#6.NN'VT!,#--".G".
M(]('9!TAOY_;ZG$LC[=X8J4=_EOL,1<8ORO8=GBTX:/BKS#UWY;L/_5\R@2W
M[B[G2T[4;I[U?J1I^\=<0J3\[0<]CU*K=KVA&V6D0U4-+&9Y]RVQ6NF/02C,
M]!369JX&,N@?4+ZW-H8L[&U[L"GHW=;.9O$;P(;+E=OF:'[<[K"[.OM^M_._
M"($5J/Q7OSR'M<P7);6Y7L$['Y0($YP6X-UY1.4/4R9+<>?T%*=C]=",A>B&
M#AOWW;247'PO&(OY5AV;"#AR&5RV0RAUJH#4A8^)&RE`1<&D_>1C0(/^M1?P
MP`I][NJXHD!5ADG$?)K<`!;DY1,#`E^#,MVK&7,#4TIU2RT."$A@9OE,UW5>
MKMU4Y8VXM&#>B3AM3']Y63![Z-+KG?\>]WBY?<4G23'?,X$%)=9Q(8:&3VPD
MV++@B*COI2\:Z])UJJW.CYN?G,(PLI*^@6,C8.XMQ3\RS)!DS)F#B3QN:C,O
MC^O\>SL5B8)M6I#]EB]YCB`@R,PRT7)F8^S1_*+20Q[B0:6_IV3HJ;-:5AM"
M;OLX<O'@2U)@<%K%'?FF`GJ1Y^=1JJ9[AE;1=^&/-)OK8?EW9VK,I%OWI)ZC
MP4X6?*>=#<GP"W+XEFI`D`V9[!`DY.]P,<5\@Z7Q27IP%O`&]D]:[N/[:W-#
M)["N22G^(5K<,/=*<)&B7L+09YK??,<&[_(5(WJS5XZN?CV#*YC\YM/XRBI>
M)0M\`[5TT=9]6\$^NE4GP=F$797R+=6=<FQ;]7DZEH(2T$_:PE#TL`R$)C(*
MM^I+LYYN)2"V*A+Q@,@+)A`6,T0AM\B:\M%1EN=V#2[*PBNPX.\?$,(GG8C*_
M_]*+4UZO)USTYJ.?W=FQTJ?,+US4X'*8=E?4E$=5L_^CE2'SU7>@RFQM-@DL
M9]"04\D`!Q,KD\8_?Y1[DQ[&*U`%6>C$9#!#S:N\['S;R-7OOW"^_OBHGUJ"
MK:]CX)!7O>#C8'R!>`!!W6T`?S"H@IF^S/-B8E-$5^Q;23^'F!;D_/)]9*94
M"C.3Q6:]3DI'1<.A,)=%7\%!9\&>"&9LD#X][OJNKK`3HL;)P:F,+Z#-S%]U
M=L!W[1=B\"[S"DF)K9!I//NS63P1(>6[+%"VK-&OTV#M5GU8X(),WPPR">#S
M9X=9U1[*ZW?^HV*MC%O;`>ADATD;V]C'2>J;3:SQD^1S4;3O)70#SOCH^Z=6
M-@X"YA="`Y$I2K^4BQB^&7;]=G%BJ]Y8(GSB0Y$@R1;N=`#;=H:,MM9S@`I-
M(#UVPP1/O4U7=M>AV5JCMR"NC\3X,-+D>W2P\>B5SL-\6D0Q/.AXXB?W?*OK
M(,WU+#+M.PE:B<L_[PO+*]F^L"5/\W0@+-Y3R*2&L=6R;/`U0IFP0THYJI'E
MZ;8&GL1#<*\Q`-*]^G7"+E8>][GDASLX$*^M)^WF)QUAPBHGVL#BDO78616=
MDH4==CL\TH^&NG,J=]5$#8(979T>];TI:7B%#A4_?/;^:W8^UON`CV!X4<I=
M!C9+ZP,4!->M/I?8\W.*MO@WWRK&;,E"5!#;]4O:-9A:B3)-K-"RN-(1F%.Y
M3!N0P!KI+WBO;@/_Z8_K/T'=^;=3P2`P%A!CO5@(#XY-'^YQVE77^-7OM/]+
M$];G"M)Z2Z\I]GT<L9L@/_!PGRSKODI(Q"=`?0@`2;5?^0.7H;VZ2PR7DY1W
M,?!PK$E(3Z;)O/WGW<O!EX;`_]I^456O;OKQF3>_6J3-W0:295:8:59N?+NN
ME'%?*<N.3KF=-//ZGZL)#'3><]Y0__CM^GQ\S6*#>#3W2/X"'E'/J#*+2VL?
M@M8(57F[S">!6-DK]O+I9Q=0T&1N5DBWPO/P0.)G0K`[$>V'N`NN676.FR+Z
MV_*5V[]A?L40+IN+*4Q<=ABIU""D0Z;S!A0(>7Q7E7-("FZP-IY5ZZH)XNRR
MH"4E[G_((B^9-5'W76HMR*(P$B?[>?6)]3VFEU>S/9SQYE!@@':?^GNUS^)]
MEA&^#ZRH#,T5?.@]-PO-FG3\Y:4G<KHPL',^+SH+=5+DB*?:52'PKA3#E>^5
M$CTLP<V2<%M7^4HK,KOQ9B^<5>+8F>&[7L.+QT\&M2.]I:)/S21!/ENE8NV+
MXFQ,3^EWGWB</+SYG=4RVN]-@^UQ4%<],NB$N+F_'C.&-'FMC-'YSR;BU^:X
M_^O;7=$-5/5?TRX5'N%A*&R:;493Q,%`)0LFP.R5FR#TNCVP^H(3B(SR'1V,
MSB7ZAIA)7H^0`440+0O`PV$)::$TBA?%7Y7#SQI3XCAMF!NL.X;S!:T(/F3>
MVYC5O[R6#LR*JE*:.DCG-=JG9;->7D72^J)</.$I/3L])$3YTWAF>X\.][H8
MSO9LGSUN/[5%=[U.UO#TB>)#/RYKB*]JG&=G=41XZU)5QG>(7SZ>'W#C3&V2
MLBZ:I0M[N3P5;_3.\.)M;4K,ND+TX]#PSR>[OK-C[%A*+/PU%3T1=E(DOL&>
MNPGO1;%WG(AUQ:8QOC3EXP9G[2\PBM)1H]C&PT=V9I#Y.%%ZL*U&#`](>[;_
MPC'-KL0-A[:X?-6VJW:()5KN/L;1GS1S&@W^^[T'^VL^RNFTO1JA235)SH8#
MSO-WD(?FF*:1(46CNPL>>1^=XW)(CJ"I#O607'NWVFKX(?1^+C;`+<2&1[[[
MHD\="8/!S"+WW18^P0A^_]<,UQ=/RG2'-WU,ETXE->X08TDG47S@L8F8,PL]
M`IC@_+I:Y+*R]3IV4O_.3W`1(CN$W+-7#FZS.NFPS*-)-2F=Z.C_E5?J[?B9
M-W-**DZ"3Q0?+Y"UY>S"Q!>[S/OTVY(Q*-+BO&2>\NYJ\J;VZEQ;_=*_/>>K
M?-/BK,MASFE!<2K,R0;];-Z)#&3I'XCF1XMJLTZZKR\G;6Y6?H-P>VM.30TY
M+#]P\Z"F)\;M8/T@YA`.87P(#7&_.9I2H8Z>E:*Q+M.)[@?7DK7=-Q4?+&RM
M/#'1G[:)T#X<"@E,2?%^N6D<M@8=;)JE>>?X#:/Q_#.V=-J^Z!#T>[WLJIE-
M=!CQK>(1HR6GQ4==SSTC$W>YL:BY3R,(&F-J]NV@D.&J5]^TB`<;SP(VR_2
M#1[V8"9]C6G^7=F)J.7/5OEE+V#"_08C$W=YDC!.YF^V5$I_QL.%)&873IIU
M3T?^,"5MM_P^^!/KNWX]NJ`Y%'PMP`K<2B-AHJ:"+5L<.%=5;R/%P7&WU+R=
M8(,AXZS>$T)7!,P;Y'/BX6A?-M]"U*Z[_7=_5HA6^SF("V6)XFL%T]?BUX2<
MFX?PSKM8V^,`^6%S.D.PP'9C2554RO*=Y*EZM:&OT^)E!D?#$?'?]E%VTZ#,
M8K$%]HG9+8C.09.5:></'Q]B#;US!CW!/R:E#6,17ELMU;N:S1[9#]D[4(:2
MC)5?.\N>F74!T(&F\T-4]D6@X"2<JF&'XVI_,CT.5V1;8%.:KK9:W)_7_R3G
MQ4%.C.YGQ1B)T%:]@SALR:O[06"563\087$<W'7KR(_\$$)-0JNA)Y&2R\[\
ML48:FGH)D/Y#BV%"49^ZA,:/?>_FI_:/V\MC+I$!3>(YRW$NGK-)Z=SNA`CZ
M<8[*J3SJC/UO;>6I8*1T?R0I[_19OT.P+#A+I`:/=SXHF"AO[JLX6K:0/[Q
M0][G">.%M+1XU"Y@!2W!Z0/9]TY+:ZG0Y;_=0D\&M<34YS?86T(IGHWFH.7+
M?]!3K]^[04(22(\Q5J'T@GDN%R0/)!@[++<"IN8W[^2H=>RH_`B_.&;D.8Q*
MI5Y''MA91<P?@$^"%%8:;]E]@1[1%>8C:9"C6\DKOP,5TQ4,R33"]C,JT283

MSR+U!2&POMAIH8OA@D6_4S,LSP_BW0\(IQBI\[APV(X8HB@JAMMO$?'CN0
M<@2N-:2(,QAV&9K/;79TTH8EU54WGS,="DQ'!4S5WG9>T*.0B8M55)D@7<-'
MAJV&5+P*'G#H=-;.-,<%Z`]MRG;\[_6"JCB%B#=?YSPD(-960W=LN=<"UBCH
M=AS>-6.-?+=.`C-HT-%QC%-OQ:;FD`N(H(1CW0V[0!?S2]^VOB1W77Z$1Q?L
MOU];6T?3_H?62EB`E>2TQ63`?">\K`"COVKIM>5G'6V[J<T0C`"%>RIV2SZ.
M:PA&5;>CTX%3_\&P0W+?[M,CD#%I.!L]SA$,K)$JJM"UI+197#`L2;K26&A'
M?^]]9-'KNUXR'S8+$Y7\7>1,S\&UJ`-_*JYD&(`ZO_62:WC1'&[P-LU'ZN3<
M=D.QB8)\&T,_L=]!CGBK-<3\&#*P\R8@N\/]0LH+SFY"#T1[%VB%$_*VW;UB
M)Q)]J@C8T4;5).HH^N2$),3*WPSX%1"UAN61S6"6@$32%KV`T>[.4K/M'ON"
MQEJS;8?JO4_EM%W9ES<\I4'K^01]KMI.P+T^5QJR6^'M":RI,2&_:EB17DPB
MT?>6SU+,ZST[;JX\!2&@GCFZI4]-)0WNR@_**^$O[SLV!1X28GH!W2$U?@[2
MJ]<*#B>G[+]5;`6#'$J2^'*Y+CZF*=!(ZM$&?5J(;9[K<UJ"^3\,'<KN"`ZZ
M-ANQWRALB[L-,*NNAJ<)_#OP8W?EB]_P/;G[]`?5L6J')R?<Z$2828B+@<[?
MT[UJQP_'@2]G\0O(]F(*NK^\:*B[O=.9HO_=([$R.2=-2D%OB%$$IJS7<?,E
M5]AA4%+OVBJ$UG%IU2D?YZ^R%!?#4N;[FHDA=P;)YF\HW+7?Q/#?K=,[8F95
M7MZG1>2@8<J>\SG]%J,<$<[Q,/+J<7]/)>X`\:2[R9SBAF.^Y^%X"7",X5?^
M34LN?7X<H'@M*YCI"`SFG![2<S[/9H?DO>0.7E@PKJ*%&]O-#!+1J>YWMK2_
M+?3J_9W\3_=LXCR,R&KK\[QMK@@.^F"!+75WZ&M##^$M(W!ASI9-0B\RSDE
M#?\PWIO-^P.#*5//A;'SL"JSG53OM3OW=HA:5;+'[*Z4A`PIPX=`M0V0(Z-7
MYA+J'L0E,K;U2FL7P5\(Y0JVL#O!L[WD\-_N59<>GQYSSVPV\9I"6K^%U*/7
MS]:5)P*L*S=!RW2K0`9*&S_4W2AT\D?!<0"B]QSE4;@E(0DKND^`6^<A#70V
M!K0)0TT2J&WH8#FWOG0$4>I1>%;H!8#@7&#/[)O*@+8=9SL_)3*O?]:B*`3>
MR$TE&$9EKT'TO\#-^!-@6'AM8'1,DE#&AP/-I;1H!O(V])41DH%.2_IQ&RWP
MMK<)$ISC0Y#&%[0#>=K/Z;''`)X?>>=#LU#Y:>CR6LF9OZ69]T.QESE'\V3E
M#R'&'YV#\WRWXE?%]=+=?-?>8(43D\EQ1>+LU-XBE^,F1XXO7N"3R!\[W:#=
M4XTV;?$#%T09;+.FF;JQH&?'N25AV(EV@SZ3LHE])(PWPERD?&HJ);YO--)#
M7+V=UK-]W':(%"F%)*UD`HJ1"`6"3Y8EXP[]O!'#&+]3-D0K/W?\:^B76?G(
MIG14JJ;D.68SQ3^%TW[1)1F+O^"*)YO7W&%3-;[:>,&NE&4I>?ZN=P_*QR]B
MC+`KT2/R?1T6&8JUR<.JA17MA:ZN?D6GHDZ6DLX?-1DQ=3WOK/-"_+T1^*FG
M=.9!OX$DXG#D/JT[^Q:^`F0S#@/MDV\ZY2?%M/?O%60<$@8:9LV8VPWUH(@3
M+)]E&3;P`6_@@KM,<QN-YSY)6S^=W#_:-NRD4G_M=-I<Z?N@M^K.R%SB7>BT
M6^W!<7A#V6CEP_"F/:!;+G%91,B[@BIM6L$K#;L$#]9;?2_)K'U9NMPT+^;U
M"G(AU]MDE.7B?'BUSQXG^Y6N2_'<XVRUHV<PQCX,5(`Y2#=*`CU2SYH5Z1\3
MP&*.KKW+9U[.+(_J;5?>IOX^!?]29]'[6+Q[?'V'OQ[:J?:[J*LZ()NX'[YU
M\G>2\7B2NNF(.&O.+V_]0L/N%81.\,<:*C&5?`R)Z^/&F\!FU7M_<7WQV&_,
M@K-W,S'(J.#@AL=#@1?M'&X(;,/;^22I!V>UG/J&47\K3K/A(S21(^4[F1.5
MZ.:;EA;WX*WN^J>JNF(*1(]D[0@7DY)<!2\!K7=*GU*K?;$-3[HZ_JHBQ(AY
M9E9ZL)O=Q^&6DAXG?ID5-RR=!.L3&)99VB+YRX@_(=SW^_W=4:%B7/*"+`*-
M]^[=K]RGS5-8;6BB7\YK*)$41OJ`;<S19H.E!U1<GC12H6>Y38X9:4EHGO5U
M(/0+X\OU>\062C7M9Q?'B/V&EAWI7%!,9ZZ4!,J"S@6F?;7(XNE"PI["!8_W
MOT\>O:G_'(=('S4P])"5]"O)YO"3WS.,QZ5_92FY`F[`]LS$N-`T<NT"5/9X
M&L0Y`Z*A6QFJEAS-]FTGCS8A$H\/*?$.0\$S`MU^0%XXE:$S??I5"6?/6!2-
M5+K]D+<F7+2TM"C*>=1^J*?INN4`U"\]2?RW']R*X52P6]SACFKB1D^2LA1[
M@V+>QO+^Q'$HR**_F?64TZ(Z/6:V$_VR<:<R7(SAS(/&='LN?BRCG7'!M[RQ
MX>&_*^S6!-G*NL4XR#D3YD]`B9)<-QWK?A>ZQG&O#A<RPG'\HV`B/9Q*ZUV0
M!PF8LJ>WG<S2ZU61GTHP.^Z^>^1>G+Y=^+=N)M'A;K[C,5L4@R.('M_XP6(H
MXM9"LI!_5>2^8UWTI[2L;U1\D^_PV=FI,-AKTCL&9V'_*18-I?B%UEJLJ7:'
MYYN'.*9<^(I%[`WWC:H'#5#/Q7O%3^K?+Q7?J=C/2\(Q]I<8HE5CTV9N)>C]
M>`G[04V0#*MK.<%ZR)ZW#]1%=7G]96_;<*DY6/8KM06>B,Q\JY39X,_Q^[OQ
MSC!_!V.0?_T7(^FR%D-;ON-M<#NQ(<3\TY?F)&2OU.6N]E[!A[24<!6J`2Q/
MT=C*0GB)'MS@C6Q&:[N9ZV/?Q:TY\.3M`^KY?3Q*`IY$E+0W+<ZAG\-*R<_F
M@'`:G[S<KE@[$XS;=Z<#_L^B^&=K%B"\BWGHNFW>8$:HI"CU/_*<Z-6P;,]4
MB$$%5>Q"-U^7J<\I:!65WW'MR<"1W$DR'488?V:Z?6)794LS9/ARUU`I=7&=
M;#/YWZNQ<\?SEMKYA*Q<)[_E]#%\.,_?*E3-&];C^G#'%`RCG'/)OM'$33\9
M-&V8[P5Q\3U0'@=2SCB>V%62W\LQ2EQFI9^V"-+Z^[%5G37"9N@>4P&[G1;F
M=>!2G&R!V+O[O<M>1*Q(H8M/IF%SP[%=O?N"95/8BMF@]3B:P,_U6\WZU61Z
MW/;Q6J;ELD43J<NUIB7_QSP[D&G83"/YN#R;C0@SSZE3-*0&T3]^3)]GE'"I
MK48;#A2ES.+H0,D5)7L\?U:UX8E/Z:FJ*$;')6GV\R#VH+<&5R.PB.<UAX)8
MNVR64E%9:.,,AV8C8O2F5_V-UK?<'-9]-QFAXB!2\1&[I?$?@RW[K,52<%YC
M\^\'+N#\`E/HZJ0H70ALIW[!Y8/^4(%CGJA](:1C"J?MTE=BVZKZ+PU/35Q`
M1'PO"\C)VOY+ZQS5K.BY5<RB%1VQH!.)LBDM0;JU,&7),$%4!6X+)O,BB\E]
M3K+(P=W^AG_NRE&%J6@>W(KVZTK53OL9"L-A'429YY/5)M"?\<%ZGXK$^5\P
MB3=C.X3)21_'>Y;5.)Y1XD=S``D*2,ICB-Y2-6O?3OJA3-1MK'W[5L7HFNCU
M]QAT<I0Y2:E%Y.%FR_$[;?CAKMV9Y$G?]1VBS[^_!$UF83_V:H``>0BQJWKK
M^/MP9SD98F-M/H4G!\,>@Y*!3II-=Z=>6QQVA5Y6AJ%^G'T0EW)4G(RL*2E%
MBHL+EK,W#.<`':LL>TYPM<H_JAY[U\'9T"KTK<`SJ_.7O"37`H!-6C9ZN_;F
MI(`!+<[H2Q1Y54LFPMO[MG,4!V<*W5*2DV%$HZXKB?NR=#=T1V&>9KZ#"BB/
M]`YJ3?VRE;*$^BQF7:M8\M9_W=!9R.Q7TJ5`2[L5"K#.R,2';LEI60MF1HV7
M`!/'"_=2]U2[DJ#SO*.#/V:;%LR;DGGI-SO*"M[N"/[2'0IBHL^=B8,@[7!?
M-91TF[2<PF0-W8AVN@S+XFXNZH0YEO;#%IB"\7IJ*INUKD1U1TK%%C=G)*;
M'T0(*4\3FWWR,,&N,7>VP?!B&"*6USL&M-!U?OY)57`KQFIO/;@-#TL2@2=:
M_9,#<P:=K@+V/'H%@=%+S282^F89&T[@2/Q8EUS8>:?^7&3^J8+U<15NI^%A
MEIO+&6*;U<:/05R;MW9]1;I#$9.5!#`6Q1_S5VX/[>KP__=,^>K*6OES%=6)
M,\T90AHS7I1[5TD).Y(D0!&[Z,*J-7[AXPK81_KJ&YI'@+`R"T,_/"VP^#(C
M.H_J4.&KK*PVU:PM<1%\Q(()[8EF$8TMR8YCP,#$XO_G8A1S0YM6AP//;EX0
MF.M,1B<XOT'A^V4N.3,E1(N!E2XV+,#!?,G)643197HAP)+3+;<9T?,R)#`1
M.RWI9W).?$98]:WCGL74\8VL>ND.H??[M94-5_CXBBJE9UU=SW^M`H9]ZD@\
M0Q^R]LP0S!T>G12CZ*"Z^>",LB[S^1I[=>F/3^S19TGL_'[H^.QPBX[?A!=G
MX^;C\GRSD)K,()65'VT(!!2^?EH?4=$NG)B+&AL="YEO4?'=M/*/@$S4Z/^\
M<?;VUK-RP;\'C$:;?D?#MRK1K#QM/_/>0.ABOD@<BMQKZRG9^>:^!?[)GJYC
M19KJ#Z @W/)[,(-BP'>X7?*F:G[*JJD53)0EJ=D(D1PP<E7!J,27 JL`=SBX

M[,_&!$3'?W2&66AP.]E-;<@;UO$^379=PNM/@DYF=:^@-P*[%7=E1>$F]"^N
M]7E9KZN<I4MX8W9M+(.FUQ^2LKJ4S+",^E@F2^933C.K$%TY+>K(V\LSHA&W
MD/\RM;_<804`UGQ1>627^^3G;_(A+EZ^ON:[Q[]P.7E\JZ.:AJ$13+C$ZD:3
M65Y1""+EUF&S:,^F[2X=QWIEI4+RLM=("9C4HH,'KH&1W4_EM>Q^_K-V,*]-
M,B")])PU,RI%;'@M_I0IW!`A?KDH\3*VLWU9<RLYB_C<<&1'\679L[$)J`2L
M^_7CPS`.]T5*''[;E``5H[`_UX4=*G^UOU]N^+B!6R>Z5W/R?@HL0\"DX^ZY
M'4YR947:W#`'1#!2IL-,G]TVYUM@H@!$KA5)/NSJA7M#;\.A&WUWZGG7(FC[
MCR1P:$VF`4DV5N*_$A6RD_+L,>'IKW7@NXDTDB^[V>:*&*3M;1X1IEZ<+:XZ
M$Z)<>BJF#NZYE,&?Q$'L8%#];>J6`E:<I1&-^^+*1:NZ'[O&4$*04<?SYCP[
M;L+_J;\LIU9F<[DWSYXL8:$/&Q%FB4F)N[U8C09((90BI@[D^L)]%>38V8!/
M70*X$J;+CFA^.R%HI9UQU[HWMS7L*(X345"U5X;%S,I3J?;G.O.]F)S/#_X%
MD(@QJ9U_*JJKR]7T2'C=)GXPJL<;7NZU]R**V!RPU^XB[Y_IU!]?:>7^AE+1
ME:@$T:,M'$S][5)U[>T=\3MS>*6D_-%J^XO,52-^WVYSUDWN-A`YP_3'Y4'6
MLV/LQE@*[X?ZV7V5V9=$*:P1LHB(MV,-ISXT@E2U`BM`ANVKO$^/J>-J="9U
M?GB="8T":8G,:!8J]6..,(.\.*&*EC1PSP%CG+<O&J=ETQ@4@&[.]UWA3;(=
MI>JL?J@[<:`48@VICOD>1OG=;:P/O5W'#P].B<7:6D:[D[S3$C@<40.0W&&R
M)"D$6F"L(W2[K:RN&377U2(4;B#HUTFJ*E79F);FDI=TB7P@)KU9;_WN#5G_
MD_QOX6A<!OA1JPKRLPL_%^3_Z*5/YJ)E2`IS3#YL@5B?/Y(G]OW4K'EGZ>.Q
M7I-^[+4^VB]+APMPQ?$(V6U8$YHG\#^\8)+8:)#^>=O<0!#'[>/(XKE/&HOB
MZ;+(N5KFA6B?5&-'(Y!SDK3;T(S]-ZUQ;Q=<HH1-R2<%>,[^M+A:96#$BF04
M"VN3O#6_N7)R,BYN6SL0Z<1G!F#LI\UY!N\9CTX02W?W_^R28WF;A].62(#C
M;3&KZ*)668FT!3O*D6V9PA%F**0^R_;KG&*]PY.BG/,]&B\DD`(Y``%8ZP<(
MVON+ZUY)`J\SJ81`AYCL_:VO%+2QGIT@%7'/>:!:P$[$\X/\^MY8JZ(I\NW:
MPIZ@^C>IC?/ND>2-^Z=,WVXQI!O5XSPL,2X.E_-$`HJ8B5II6M&[JX^\"*P5
MG"5J,S_5MI:'ZZQZ(,2A'9(5M[-](\F3T0UJ8EQZV>@]%AYT@38U77:.!-'L
MQT3Y3;4GM3+W3D;5.%A\[`M.@%]BQDA]3>)N?E4^SQ>1R@I8V0(>OF1UQ6#Z
M-N6FA1Y9,G$>.-R69+T6,WEMNC2;/)TFK77NMTL57'3DRUG]OEJOKI*X6BDN
M!;9EW6>]1\[`651VSM,?#GS9VF@+4)K?GOF'W3SO+L[,!CZ\61>/ZXK@K@3H
M1HD[6\T\1Z1_?FQWV%J@JJJ;@FPN\(A,R.^?H+^3Z*R[QZJ"43Q3TC)6<A%
M*[S`V!>FG'X\'=09<&Z+$ZD*J,(`&[-*8J)*(KCL9[F"3^=>L]NP]!8).Q>.
MI:;$[`D'Y67DW0J\3:]9XAYV"9Y]0U6,*SD1E(KU:'P8A,)*XF+C)V-(J"._
MD,6:;9:*JQT.)[(#E$J2!=Z[G&:@WG,%#/F'Y+R0_F0E;*ZITQQQ^"Y,E^DQ
M5YNX709;S%9OZN6<AF1O2KY>'M6DH>!0MB3K'SA1_28EI8AWF&-0K-#<)6@)
M<(/'!U8M<1'5Q1'(T+6O;=#PDNLX;OLV.NPG,W]9:)"C]S)@39BDI;7&:I2=
ME)2:/@!VRFRU>Y@&]/XPM7KWAG`XT.CNAM\5/4E:=7/GN_VYCZ'!O102?IH9
M(X97!U'G%3V#WR!BRHX'8LO)VM!:37>UT/`@T&`D`9R:@NMVD^6,RXP9.1E:
M5=SKD.&%?OX>7ZB';.V1MI;)\QA3PSY6::P=C3!8R'"XBIV&L6?K%:8A/[>'
MAKPM)29+2PM!%0=\/B6W`;WE02<AQK,_G?W"4*AERW/$@*R!JU;/F(\D"+>3
MO4\'H!;29-?_7-W/M<0C/&LKK.2J6R=KO/!N'%H2">BSQ<%-9CG](@'ZBZ#+
M>P#6K7>Y=+3:O$E;RUA``!=(Q*U.V/ZR)\+1N!YK==2:25ZK,PQ:]Y5$C1%9
MU@W*B?+Z_EWU,Q\G.)AB)K4X[78S-,V==RN&RWOOG1@1V&F(AYF%(O<T"$$]
MA/H^:S&M4_UDKDJ;Q+K)YX[_!4YMH5+#KV3T"^KIQ,8/G&^?^VF'5(ESZGC*
MCJ$8.#.-++]!43NW6B\H&5ZBPKS,E/EFHR2U_23Z>1<41!\I)6J!_`HA2MRX
MX>N>,BU/T5FR,G/@8WA,1UL)ZB?>Y-G3Y2>6597C_2C=C\J4BINO8,6_5C^^
M8+!BJW+"'D6B$1M!A8XKW%/XM@]95%`8*:`<MO4-*YY)UL>_K+0PGW'5@C_:
M;'\%SR[8C.L?T^RU>63_B#CI#*^FPYC#PVIY?5TMI[_IWM2Y&DYAYX16\#\3
MD@`6_>QQ87L%&VH-);Z5SGI=>P:+(9Q9Q3&7L(H+"/5EIV<9=`T!`SDA\K!,
M^?O.GZ#K39QWKF>?]#O.3YUFG^J?B.M+,K:OC?WUFOIY<&&+7JN=8&B)\88)'
M.L71H%;M,_&-445.XNR0M^$;?Q:POEY<W'P&I,[MTE?5E2<*.>"&J%2_O-,V
M7<'H96`*_,5`I;6?^")^&34XV"+5QYH\9,8@;2;MVC=RMB=]!NF"K-,5`4T[
M'SS3:.F6>105"!DB/KZ]H1)F(&)IA>`:@2/*:_&X`F8R:T3J!7FXB)J8^35-
M[$<,8HW]Y0*B'A#(8%_.IN=JAPD,G<_/,=DI]Q!_SN8_A14MZ$\50FV`D+.5
MJ9L)N9V+Y]]Y&*7\9FXLSKXG?BH`-4"NB7+><]S#W]G@!TTOO&T6OT4X8GY#
M<KS--1TZ_W)7U76Q:<&N<T]C#VF@GFY;7=%8,^H]0MZUN7MXT7UKJ:G"_NVK
MKG6)=7MM0A<^64@(XR>TFZD97NKLBJ5"&BI:K*V_LW4EF#T!^0.%&!1KW2@A
MO>C"M8V>`8:"]6PA44W%5A4KZZ*%^%_$40$5OOMF@\J4^':]`T*,$G[DP=))
MU%G:$3)EFY+RZO*IZQC)U@2E!8Z\'@=.-&R)A)BQCLHWWQT6!,G<Y%V*"6@Z
MOVT;=6H7*0PYU9VL*PHTBF%L=HJ=2S74GO@'_O.O!D[VLM8E^52MTN#$](8P
M(-UL5M,;*=W0SPC5G(RZRT],1F,9ZZ[@VJ=R;MM&9U[6\4I>MA.`OTIH7STK
M[&4Y&3LU,(W8?>^(3*>N,/BPU[I^Y@UO>^CN*Z\Z.6Y#_Q;C(_C1NN]B5'(R
M;MG'-54(7\T.Q,V)9$B-C<;PZ;V\,OJG)A\EWW]ZA-(C#EB-[9.9A-C$4=F-
MXJZ;/OE!,2)(_R*I&S3FQU&M58BD3B[8G1JL:!&F9?8.[B4R2W._0AK%'<N(
M53\%LX5@QC=A`%RSCC1F*ZIF$R.II.7WMG>S_UX7Z@`BD'::&ON*?#?M?!MA
M?J]532'S:<0>#BB^T[7GB*CKOGZ6DBZA$<D[=*B4$:3X=72#.+*+G=%*6MS-
MY@Q;1?-NX:=X(VYET*N3`\;B1>^]"18[<;5G!?[5#62&5=S.RRNV;VR[D])W
M1=&P-?A`LJ:SZ<[!04D^*S43V7(]$Z_-R.;0<>LAJ[0+5-5J\NYL)%1P$L3C
MG9,?TQ.C0OT^5ZVQV_E+E0E1\<#L.QY*LT^;/`P#IO*F`1G6I#$_''RJZ&C1
MBH)`VW@;DX+L7Y@9G*\O?V*X0EY1<+O?MPNNG"E]VB?72%&TM9GG;,?>Y%HL
MM-";`R`%#8ZB#O<PO%7_9FHM)Z]85%PF[6SW?<GJH#)8RO'9&K-[/^RI4QXC
MW<8VD(J9)?AQ8/,!\1)B>0_.DZGDG..HE117/R5-9+;8ZCUJ$R3=#PM8@99O
M,@9^1RDV4,J`L6P>I.E6:%-34>A(W`,6P5Q_PH^/:5X',TTX@;0X-Z'>H8P'
M=8K=@DU-BG8DBO5F/QJ::R`!D.J+(65WOKLW]D\;)Q#`@I__2545N8R3T<*#
M92,3AT_D0F["W.B;UI438?;&!JSJZYZ/!$-7V`9FZDB*)&])RY'D3ZC?9"YB
M>I:,,1>HF)WL]+#LLIOL@ITUQ-2EK,C=0:1E4FOS*3X[Y7-(3Q3+/0L;"1"2*
MA(Z/%A&9DP$"Q#,(%KI0:,$"6#[P2,">E#!9S'<&W:@W-YMZ&Z5F/OP[9AI7
M^5.Q,$<BVBG=TJ?E]Z)8Y2C-_0Q%]G^WICOT%0,^_V4"_E1.O'#L@7>ZVYW$
M$<./`Y\LKWVX]X.?&?,THF>BVU:7N6/)*BYKT=*I'9+UC>!SXE3-D2BOW(^.
MVBD5.C34<`B<+'F;5(=WH<4]UO/Y_>VLN%!4]WN7N>%BH^_CN-]WT#3:RK?U
M;MIC&K5.XXLX?E.-JNX7^;FK]U3QJ$@%-09=!V?&FFX$S0=-0_B+6DRF=8V_
MF084RZ9'YC2ZE6?)JBN,B_$?Y_?8.-3E?PTO3B.%#1PU8M?.&DO?C,_E)A$1
M4]#P8 *(?.K.3@XP10HBC3V;4(G:[>E(\?M['ZS:(/GA<6]O%>NI:SI\'LNS

MS3%',]O&Q_CB7?M)%^+I.)`#ZO!AVU4IQS9GW/;?4`..V-R#PLH)-8EN.`8N
MG=TW[0.G:8OB/WEOVJZDE-V7UD),#+QM]AIDW'1B-I;P5K@KN>])4R^MYED^
M+B1HG5:>4C>A?:Y#)$L#0T$=<X7E"6]B:\;K!9)QB.^]*:`@+`Q<`C;3]#V>
M@C11P,PU^FYA:U:C9SWI<@1OJ=D\E&MZ.[VP/5E/GT]GWQ"*4],HRT7'_UZ%
M/O[9[</?'>C=2UY63'*%<0.@BOH=XUMJ?/GW(:LEAH;XF6&"A'R,>.04]J8A
M7WZ+>6YLO/YMXJ>Z6][4S0,`7G6EILC"//'T'1*<ER#3H]V_OQ=4\4)>[TIN
M9TC:H,BD=7E#89_*JSZ;H-W9>1>7,=\/!D9*(`ARY[?/S^A/T=Y\F7NN=L6
M8:Z1KK0TOC/`$UT6XX2$*.C\;575N\&)DJMC^>STZ`?JQ70\<15KF!P\4)KF
MGZSDCSFJJ.'E@-.HRE<C%>A]4OCB3/1[2E2,G8L.>1Q*F/9U[/G2IMC','6O
MI4KK[]"&L7O3,4`/]1*M?7-W:]!&&]FT?`UP?&88CAYHGH3\W%(F4IZRLI=Y
MU`$'^#3A'"ZQ,:R9)"8`Y&3J(M^\TRM-#\2H!5?"[A+_S:9^-62QM=<BW'PA
M(I?@/*N=I39K;21E/G*&UT4SY+OF%VU1Q"!K#1X(;HZ\4YLB;CGPS#)(L\7>
M+UE8U@-#4]OU.QR.[XYD\F?D&O1,UEE-]Q"/Y#9$/[HZ!(J:4I7]<HR]S=RN
ML^F-UPP](N-D^ZCWE=>DLX"-CAX7L^6Z_'KN9L8D]9<">JJY3=@2-#L':UYH
MK:3M);IB_#][`F20(F:O1S70(\]6R!5B.4V]+$`5@P$@ZW7>JE5V/DZ'$RT.
M'\3MNB1$^>P%SQD,35=E>Y,EAGB%:P$6BFI;6HU\9F:BF"G4<0D)-"R89&5[
MQ\7M;TO+M(6DBD+9`9B4_\5"/7/_PMUFU=K&O?\9N-Y+/4_&!^WCI2L&@*Z]
MV3)_1]V\YRTP^JAXKRPCI3GX5U@Z&2WVVS<;`B8PT(N-OPD4.]FF([3`SN>H
MR!;9'I04ZR_W^!C87C!2=$S##;+$][^2XC@;O>WLHAE3I\`XJZ@+4<5:>`GO
M"BU)*$O0B2$Q$6"OLL2XV>]9-ND$RXU/C=4258YEENOR:I!5[RE$R&&3H:/Q
M8PKO_AA86)XW;`3<MS57*LYB"4P4GE6>K"Q3E+O[N':",J192CP^[%M._#`H
M$[&VLC8MH*1X7T:';J:(BD2XZP%`"L!@G^FT//=QKKM?&OUJ4.K^[V-TJ_%#
MFI.B1K!\:H:@"$M(,A&1K4:Z.A)ZQ#<#+0!>UWZ3^14O.2XO<6VB6XA*Q=4L
M6LC"7>:7H!E+)#U%,I2&+F\0XEYHKM#_(C/#NJ/6SETIQ*1*8Z:S,05KB+NX
M_/4@ZO[,(#4M11NL&]<+FM4^L-:A3[N7I\9?%O\S?PT=@'V7^F8C,;D7X$8Z
M^.!WV7:=;/W`%P3_)+^HI<7.T0C^=%DO$)>;:@MX1XB*;ZU>ZO)7T(MN-'5^
M8NP<P?)\?D'%`]70%'"Z567O5SYI3-T`6NXO/\<(65>D2@3]W#\ZL7NY1Q;C
M6#TI&3KOT!7*1WA\A6@E-0`;<@7O.X^-`UYM4L(2\:/L[I>;DKXE?L;>Z8D+
ME=VM>>GL0@T8R4A2BWOW-SB/K):X5!'5$+/\%W8I=EFCI0'L1;U)4`@R];4O
M2.D,!&<MW5%$L[A[(/M-;-DJ?6O[E^C?37#;1K)`KDGDP-<X\#V%X%%*A\#?
MDY]QG:LE=#H81B7W8N\Y'@^3C"F?JZ8A/%+BAN[F`A![J6;CW59..\@;=*<\
MRL<!`DBW.E*:3`QN"#H2*UK0T(7W"Y5SU2VPJNO8QWFC6A3N.A':JE84RJ0L
M_\:Y%)@C9>:?ZSR17_RCT;LD$U!YK"\0^(0<7N60YX3,@><UQF_776]H,[6F
MV<#>L/\;;7-:.FC>0'=MY:7+QO\$X$Q]>T^<#B4G:\@G1C_2M-MA*E+],F?N
MKJ>,R<K>%J<RDVB>Q)C/961\P^\E!]B`O"3J2M6(NU0LZ1VG2(75*]Z\2O;P
MZZHQ.UH7:GP;\-*N+U,,]HIGF[>3AN^^P2MP;XM<+/:F20U?N9_XU6@KI$U:
M<]]T>GR!<'?^[48C.:_I-=MS[IC9V?2#?O5YHO,0(_AU-[\U(B0SX8GO+6M)
M>2J&'"<J5O3']_S<TYKTB[<S!09["Z"NK'N>B1IF9JW:T?=*U(S,E"),\12!
M>=)')FC^1O*NM=!['3&4)ISJ>??L#GZ)G*;E>Q:SEL;FI4Q]$C![PZKKK$03
MQ37X_@RR+*#$4>>81U:3)[(=$"+SO1A*$:].C)9LA;<^78H\TYV<&/LDBD18
M/%XX8P)74]J\7EQ)5#(MT#<L.J)B430_;=+%D%:[:W]M%CX6!;E5-714]!06
MG![9P-H!V`%4OW[1FJ8P6#``!!9&%0S#_]*0I@=U"B/^%GGW-RU?+[G^J[]N
MR7G$^5%5)&O>;G*:,<_34;6B_UKN%^C[2ED&W]1>`AZT-'%VN]MKL'%-K"?&
M.Q\5AB"1N?P(;/59SON>SO_F#<ZUC/F7`H;/;W\$Z*T!I4"()C:&C:L?&6_%
MF-G_\F8;8?K4,<K%T_T.RI<-^1`Z6#2M;8LMW/RJYA2PKUSGDW7:H,$J`/6C
MGIG0$5/Z9=*!G`N.7,.`MDZ'V>B\-<@L3C5V-/8NPB'"_>5R&3_R$0GY%U/!
M%M-R]\RZNFCS6\$/#E+XND51_'^B8U"1\Y^<!.:2A[KQRF#ZQ0^Z*'H796+S
M\>_:Z#+_WW]<5>;S\-F#=_4S[0I<_>I2GUEF+LAN6R[\%0\FT9&*%Y`#-#.
M4S.X39HL;`X>6@Y^5-I^MSOJAAK859#UCX&)LGG=K"@R))V+<@,,("6&B#K3
M/H2F)A&>\Y"!C(.">WUW@)O0D4_A]X`@4QD&QB="/-K8[$P4P,7">FOGX+49
M0[J>5GDK&('8"`?\?`)=/)2)W@*Q:HT,=T&]!"KQQUUGLU2KMJ`3?:4[VY4%
MT5#>-:TVXW2;S+*Z.PGXZ;=_ZHB#"[NH?B^]Y:MR:H&E;Z0JO!D?F"E>5[9L
MXL+E@&?ON=>U;4!-J1M(FWYC$?&!?Z5S1@@UC7U0KJ&_DSR4V<S64[O3E:$2
M2'+ZBLM"][:D$6Z#KL,\/5:&5,0/$U,Q?#LNQ:_:%VL^30\K]RUX(3\R-72:
M[G2[%+]>%>NNY=N4./DC/W8!@QQ3.O=9,`13B80,<=AG9><KP]Q_]\1-1)Y]
M?9D\(4%#BL@/+E,L>!E=J-_;M>K'Q9=/5^\)Z/;ERT.3F4;OV;;+;'4HQ&DR
M1.LN*WR3J3%=@0R@8T5C]J%#<EQW<(.+[OC9O7?P;9BNO$93E\&VF<O_DK^<
M=ZN3H>]A>(+)A_+0T5QS>73F]MTPW<?.0-W,G6'%B#0'^?LGA>XX^,W<S/C6
M7DFE76>*Z=G(3LLC!'5%U_!(VR>9@C\L_#4C!P\`0?SCM&V:JW.%E*\MV``L
M3!OM2'>HTT/0%6UGK[<>\E[U$$YT%Y#43$>P!GN-H5]%NRA)!_2>?:-JH5*J
MRKT)1%+3+_[A-*0!=Z/%>XFBLI.XECT,RK9?5R/,`30>KU1LP((!3]>`>WXY
M'U78AY9%FYT0]VG<:V^-I,(RYP@HTZMT2B,>*-BN\;\"G]DB_?G_T1"$4/-
MIN#K"0C8)5`7)@1(/1USHCY6::OE`9HK#>":`I#5\95UB8[3&UY-OIO1GV)=
MC2B`*NP2"Q/K(K)W_#P.Z3R,#<B*OS9*?X>SOZ=X']#<@U3P5A473YEAY5^'
MP7=W9U??AWF;'Z*#XE.:S%]5H,1!54J(=CY*HR%5E[NKA4WSSJ!81+`8%-=
MA'<%X;(CGKN^I\2NL:^#.Y2G82XD].^%(/0-C0L'_$FU8EYL8)\C2KU,QJ/Q
M&E%7![]H#!"`,FFP*N'2CQF?J?O\PTS$SP(#,O:'9/S.77F[*9VXLM.3B.'D"
M4X8#=1%RG"C[:H9W+*CX.F+NIC:V*1/1RBE:M1G3Z2'Y\7_5OY^=*0_OK$F^
M83#]O(,7-V13DI8#?LJ?CQ3@A/*3;_((U1_KL@]#<EX\8`<,%,"U1@?_V,`2
MYH<^#7D)+\9KV]"R!@G;\LU=U)V@_F(&<F4PW*,%G?SG$PIVEB':C$&2-,16
M7P#XX2FPM<-]3UKGJ"Q]?A_9>C_'..MS4`"_2'&J3LV"R]9/0*._W$L"D'W;
MVO8C!DHJ!=D!#XJ&9+\\]DIO:3R#`.H_/'LJ9'0Y?&K',GC<D?6$Y6(H!KS0
M!BY0]2@U46O?7SGNJ%.$_,7MV"I2SWV=J>[V^OC,2>_EZ21F0N5C$3\&X)3J
MUE8=N.06+)78>*V`@EOS%QYU<8]/'Z1*_$TV!)V)"E>Y10-)AXWC+Z$`<.WT
MZ0SN)#1?WM91G@*XX.EPJSJ]OCVU95<"K*]$K<`SL:W+5K+RQ"1?1]5>@05C
M;`]6J@<19P5(X"0=RVD,Y#L6&#;U$^@N=O3"+[\A-F[IU)6SRN#.VO[S2>_M
MLS+?1P-MI=(J-3%&$(/(!T$RCM_2=$R_ECE*BY00^_#N%]LSR7G3S2`&>;T\
M^U#I[$W<WP<#^/^FPU$QGR_,>OH#V?%G<1TZ:8,6KL;$?(WG5VWN<,3>&>),
MM':$DC[N:;MROS;=52DC;+_E5'/?UI\)5)VV(8AUNU*,O6=J$_F.J1JTOJ7M
M4!RE;\Y\IZ4PM67H9@RNQ57N3N%KE"3J&2H8"!L,4(Q0`YK]\<^O5P0O]T;U
M0":?>I=U)*HZ.(E4STB"1ZY'?2`YW)SV;E !&P\OOY/SB.9ML>C/6 $UNY:E

```
M-OUW,Z>NR?TG_/I?0=KFEW5Q!*NA@C27/R6X[9'$:7&22-Y/EPHBR5``?U5Y
M&HL1XP>;F_ZL$8!W77J84GR:S@@9+$X6C%:L^G;RK52HP"^8WF\QV4(ARDRE
M;;K5?#96"J/ZV&.Q4DZCCD=@[%ZP\EBFFR`NAA)>@7:.P*L8NB$$PC%`=WMQ
MMJKM*.H\`CVA:S_(PG]\6B0(X,@RQ_:JV?K(B$&M[QH;,P+ST,\*!5Q2S"G/
MRU]1??K,*:"`,5H\4+ROV'S'XCHEY29FY]1Y%O)CS]$GJ(@@[ME=H99HJKR:
M]H^:;6:QZL!\+7RFXGVF+'=YB'V&SFV6+M/POS(;#?03!.2U?]+36%$/^F\*
M4M[[6P?-'E\).IY=2V6-B6M7WY&2.5AJ@X'YJ8]5&"-,.IZ!:!/O?99,B9(-
M,6-/81A1*C+G^S=,%4@]Z!'OWWU&PXV(YP%H,+P^ZY"9Z&3CKV3U09ZZ;`O$
M?ZH8E9]<&.`T=9%5ET=B:XZ=5G7.D#"M`CL]1-FG:]S'C8X'\8D,3B>?:I8J
MV`)45M1G<;!C*W_\A0V&L9!F.EM/;Q<@D7&SIIK89WM\&1N*`##X,FPF8G$9
MH/:^MEI`8RZT+[5[K$K22%X]Z,+!3O[!CO[8-8T--`?P<H,0_A?SC5!MUNH%
MN,C9K)LQZU;,F.V'\$//OTOT!_\8Y*U]B!ZUL(2\MPOD>JA/*BX!?+\.EASL
MS':6L<>(JJA@_,H5!N-4&T\;49S]DVYMU16C$8P_`$$A;[^09$M>1"'):MR/
MY9O9O,_ZM?:#\-V(/B_NK^_RL>CR@1,4;PR1&%O2T$*9/R!OI11TQ=S$"+Q*
M7OW*[:<GZ./8K3QLMB&,IV/\F;N3A,FTCY-?N#,MO=FGR(Y9<=W9&E94C$9*
ME6YJ^F=K'PI5!?3<WXZ0,^W2H&!B801P6YC$FP4&XOKC0!].5!CL?+&4QCI8
M'J?B_(N7-504K!E*QTQ2#%+8L&6J25GR0,`T/I^6"<79O("T^=M?B;:,0[2/
MOU*[PX^T&W20FRE1L4,&T_:?A_>"3Z:&]=7EWS!2;+3Q-HRRY2/G/UR8,+0_
M$7_D4Y*=[QU4$A"CK%+49%6&)+M&HC]BB:"Q!6`%UHJ>_DA!13K\%/+<\(D'
M&U?FHTG=:$*/A0SC)Q99K<UD)#3`9[!F%1/#9HU.8*G&0\N#3\8[C.'IATN,
M,D`,P_0^TB\&&M1+G-^8#*,H#'BHOZ;O\NG_D_Q&_14"O^0;[4:N01_]#R)$
M_^]#88-:VF*.4?X_UML9H[SHQ1N$$?HR3="-&H^S"9C&*,.=7O\/C*1PP$?]
M#W4@0"V-%-Z#ASZG'P%"7PKAPW^?_\_'H[("2AD@]#GT)?*2M0QCFJSJ$&4:
M9?((9?(0I0PS9@ZE"N`3$XF!'XG!\LSR,CF',CJ]5/1<;EHF%(I41AU#-'U\
MK*<\@!$SDU6%C_J?(0'J)#ZJZ?!OI/^<7SZF?R"<GF]Y]__MDP^CU!0`0/H_
M0D9&1D$%_$=(R*CHF-BX^,1D5&__-_T_V?]P='2,_[$HJ&CH&%@X>$2DVBG_
MS5'1,##_)\'$QL''!Q#RU/PWG"__1X#_GY#0\%38D`"7X\QO\``YWUI(#H'#
MMT9+.T=\R>+OCGV"U3]MAJ.!R;57A4T+\>,9CLT/D8C@)#DP`H,2H9YUM&TA
MD21O30H>MF\$0Y,MR&.!R7?Q<NTU..H4793O>A011#>CF29:O0,!I.:Z-'HN
M'@-!I'9"USW$8&KCA6O%,0X<;MX9`<77*/M-LMJB=`/L+UBF#7H?0H??#S`K
MJI!&,T[_!I0!<@"+.8#E',!*#F"UC"$'@)H#0,L!H.<`,`ZGM_Y3(,@A7#Z<
M?OH-Z!#QR[%`[U#QRZE'/]Q\,WUA2HJ/BD9,CO0&C9B!G(`8%Y6!`9D2!^D-
M`3XW+AHA"18#*AHW$C$6"=K_`U!+`0(5`Q0```(`(%>VB9V^2?#&`$``)$!
M```3``P`````````$"D@0````!&3$DM=FEE=V5R+DY44T,N<')G55@(`,_W
M=3=!$W4W4$L!`A4#%````@`@E[:)KD%[BAU`0```0(``!(`#`````````^_^
M0*2!60$``$9,22UV:65W97(N4$%,+G!R9U58"`#/]W4W1!-U-U!+`0(5`Q0`
M```(`.0VXR;HHOKRKQH``(4Q````,``P`````````$"D@0X#``!D96)R:7,N
M<&%N:6-56`@`00A^-RL(?C=02P$"%0,4````"`#R@N,F-3IK999S`#Z<P``
M#P`,``````````!`I('W'0``=6YD97)W871E<BUN='-C55@(`&>-?C=GC7XW
M4$L!`A4#%````@`G&/'&N5*/*M1=```8G0```X`#`````````````0*2!RI$`
M`'5N9&5R=V%T97(M<&%L55@(`$N-?C?HB!,L4$L%!@`````%``4`<`$``%<&
$`0``````
`
end
.......
....
..
.                                    C=H #18

::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::


Obj3d trilogy
-------------

        The rest of this issue is devoted to obj3d.  Obj3d is a set of
routines for creating and manipulating 3D worlds, and builds upon the
lib3d routines discussed in issue 16 (the lib3d routines have also been
upgraded, including a multicolor version).

        The first article introduces the use of obj3d with a simple example
program which places two objects into the world (and lets you move one of
them around).  The second article is the programming reference guide, with
a memory map and a list of routines/routine descriptions.  The third article
covers "stroids", a more involved example program which demonstrates several
game-type algorithms and ideas.

        Mark Seelye, mseelye@yahoo.com, has written a really nice "object
editor", i.e. a program for editing objects for use in obj3d programs.
Unfortunately it was not quite finished at the time of this issue, but
hopefully it will be featured in the next issue.  For more information,
contact Mark!

        All files are in a .zip files, included at the end of this
issue:

        obj3d.o         -- Obj3d library
        lib3dv2.o       -- v2.0 of lib3d library
        lib3dv2mc.o     -- Multicolor version
        table13         -- $C000 table
        rotmat.o        -- $5E00 table
```

```
        ptabgen         -- program used to create above tables

        bigtets.b.o     -- Simple example program
        bigtets.b.s     -- Source code, in El Cheapo Assembler format
                           (see the Fridge for Cheapass docs)

        stroids.o       -- Stroids
        stroids.n.s     -- Source code for stroids, cheapass format

        loader          -- Program to load in relevant files

        stroids         -- Single-file stroids executable (load and run)

        obj3d.ref       -- obj3d programmer's reference guide
        lib3dv2.ref     -- lib3d v2.0 programmer's reference guide

        Binaries, source code, and documentation are also available in the
Fridge, at

        http://www.ffd2.com/fridge/obj3d/



Obj3d -- The 3D object library
---------------------------------> S. Judd 4/99

        Obj3d is a library of routines for organizing, manipulating,
and rendering 3d objects in both hires and multicolor mode, using
version 2.0 of the lib3d library routines (described a little later).
Let's just dive right in and look at some example code, to get a feel for
the routines, and just how easy it is to get a 3D object up on the screen.
This example is taken from the example program "bigtets", included in
the obj3d.zip archive in this issue.

*
* Test code
*
* Nice pair of tets
*

          ORG $1000

OBS       = $0800           ;Object records
GETIN     = $FFE4

TestCode
          LDA $D011
          ORA #$20          ;Bitmap mode
          STA $D011
          LDA #$08          ;Bitmap -> $6000
          STA $D018
          LDA $DD00
          AND #$F8
          ORA #$02          ;Bank 1
          STA $DD00
          LDA #$80
          STA $028A         ;All keys repeat

          JSR VERSION       ;If multicolor...
          BPL :hires
          LDA $D016
          ORA #$10
          STA $D016
:hires
          JSR ClrColMap

          LDA #<OBS         ;Object records
          LDY #>OBS
          JSR Init3D        ;Initialize libraries

          LDA #<TETDAT      ;Object data
          LDY #>TETDAT
          LDX #01           ;ID
          JSR AddObj        ;Add an object
          STX VOB           ;View object

          LDA #<STARDAT
          LDY #>STARDAT
          LDX #02
          JSR AddObj        ;Add another object
```

```
         STX ROB          ;Remember this object for later
         STA POINT        ;Object pointer
         STY POINT+1
         LDY #5           ;Set object center
:l1      LDA OCEN,Y
         STA (POINT),Y
         DEY
         BPL :l1

:sec     SEC              ;Solid polygons
:setp    LDX #$60         ;Bitmap at $6000
         LDA #<PATS       ;Pattern table
         LDY #>PATS
         JSR SetParms
:loop
         JSR ClrBitmap

         LDX VOB          ;Calculate view
         JSR CalcView
         JSR SortVis      ;Sort objects
         JSR DrawAllVis   ;Draw objects
         ...
```

Aside from the trivial screen clearing routine and some data tables, that's the entire program -- initialize the 64 and the libraries, add some objects, then call the three subroutines above to render the objects to the screen. Is that easy or what?

Now let's have a more detailed look at the library, and the example program above.

Obj3d overview

Lib3d, as given in C=Hacking #16, is a set of core routines for doing 3D graphics and calculations. Obj3d is an extension to lib3d -- it is a set of routines which uses lib3d. Its purpose is to take the tedium out of managing and rendering 3d objects, and make 3d graphics accessible to all those guys who never paid attention during high school math class.

I'd like to offer a bit of advice to any of those guys who might happen to be reading, though. If you intend to do any _serious_ 3D coding, for example a game, you're going to need to understand some fundamental things about 3D graphics, and about why the various library routines work the way they do. Amazingly enough, C=Hacking has a series of articles on 3D graphics, with a very comprehensive article in issue #16. I read it, several times, before doing obj3d, and suggest you do the same.

Knowing coders the way I do, I can see that you're going to ignore my advice; that's fine. For now, you can just jump in and play around with the example programs a little bit, to get a feel for things. But if you later decide to attempt something more involved than displaying objects on the screen, and are unwilling to take a little time to understand how things work, then you should expect to fail miserably, or spend many frustrating hours wondering why something isn't working the way you expect.

Therefore, before you begin work on your dream program, I suggest spending an hour or so with C=Hacking #16, a pencil, a piece of paper, and a brain. The last is quite important, as you'll probably have to do some thinking; if you're expecting to understand things with the flick of some magical switch, you're probably reading the wrong magazine.

Finally, a word on distribution: both lib3d and obj3d are freely distributable. You are _supposed_ to use them in your own programs. You can make money off those programs (good luck!). You don't even have to give me credit, if you really don't want to, as I figure anyone will spot lib3d a mile away. But please, *use* the thing! That's why I wrote it!

And, knowing coders the way I do, if I hear one guy sniff and loudly exclaim to everyone within earshot that he never uses other people's code, I'm going to take your tunes, charsets, sprites, editors, assemblers, and all that hardware you didn't design, and bonk you over the head with it.

Alright, then. Let's talk some code.

Organization

The obj3d routines can essentially be split into three groups: routines to manage objects globally, routines to manipulate individual objects, and visualization/rendering routines.

The global management routines are for adding/deleting objects to/from the world, setting up objects for use by other routines, and retrieving information about objects (e.g. location) for use by the

programmer.
        The manipulation routines are used to manipulate individual objects: to move objects forwards and backwards and side to side, to rotate objects about their roll, pitch, or yaw axis, etc. In other words, these are routines to change the position and orientation of a given object within the world.
        The visualization routines are naturally used to visualize the world. These are routines to compute the view from a specific object and render that view to the screen, as well as how to render it -- what bitmap, render it solid or wireframe, etc. There are also routines for drawing individual objects and even individual faces. These routines are the main interface into the 3d library.

        In order to support the obj3d routines, lib3d has been upgraded to version 2.0. The rotation routines have been changed to ease a number of calculations. Three new routines have been added, for plotting points and lines, and for determining which version of the library is in memory. Finally, a multicolor version of the library is available, in addition to the hires version, which includes screen aspect ratio coordinate-corrections as well as multicolor rendering routines.

        Let's now return to the example program.

Example program explained

        The first part of the code simply sets up 64-related stuff, turning on bitmap mode, etc. JSR VERSION is used to determine whether the hires or multicolor version of lib3d is being used; if multicolor, then multicolor mode is enabled:

```
*
* Test code
*
* Nice pair of tets
*

OBS      = $0800          ;Object records
GETIN    = $FFE4

TestCode
         LDA $D011
         ORA #$20         ;Bitmap mode
         STA $D011
         LDA #$08         ;Bitmap -> $6000
         STA $D018
         LDA $DD00
         AND #$F8
         ORA #$02         ;Bank 1
         STA $DD00
         LDA #$80
         STA $028A        ;All keys repeat

         JSR VERSION      ;If multicolor...
         BPL :hires
         LDA $D016
         ORA #$10
         STA $D016
:hires
         JSR ClrColMap
```

ClrColMap is not part of the obj3d library; it's just a simple routine to clear the color map. The next piece of code initializes the obj3d and lib3d libraries with some default values:

```
         LDA #<OBS        ;Object records
         LDY #>OBS
         JSR Init3D
```

If you ever move some tables around (like the $C000 table), you'll need to change table pointers _after_ calling this routine, as it initializes the zero-page lib3d pointers to certain default values. .AY points to a free area of memory, whose size depends on the number of active objects you intend to have in the world. This will be explained in more detail shortly. The next section of code adds an object into the world, using JSR AddObj:

```
         LDA #<TETDAT     ;Object data
         LDY #>TETDAT
         LDX #01          ;ID
         JSR AddObj
```

```
        STX VOB          ;View object
```

If you look at the obj3d memory map, you'll notice that of the 4k it occupies only a measly 1.5k is used for code. The rest is all used for storage -- a whole bunch of lists, used to organize all the data. One of these lists is the "active object list". What JSR AddObj does is to find the first empty spot in the active object list, and assign the object to that spot. This spot is returned in .X, and this number is how obj3d will reference the object. The STX VOB above is for later use by the program; you do not generally need to store every object number.

A maximum of 128 active objects are allowed, so the object number in .X will always be in the range 0-127. Since AddObj always grabs the first open spot, .X will in fact never be larger than the maximum number of active objects in a given program. You might want to use it for your own purposes, too, for example as an index into a list of velocities.

AddObj also "allocates" a chunk of memory to store the object record in. This area of memory is specified in the earlier call to JSR Init3D, above, by .AY. Each object currently requires 32 bytes of storage, so if there can be N active objects in the world at any given time, this area of memory needs to be of at least size 32*N. There is a distinction here between "active" and "inactive" objects; an object is "active" if it has been added in with AddObj, i.e. is actively present in the world. It may help to understand this distinction by considering a program like Elite -- although there are many different kinds of objects (different ships, asteroids, etc.), there are never more than a certain number present, i.e. active, at any given time. So even if you have 50 different types of objects, if there are never more than 6 active at any time only 6*32 bytes of RAM are needed.

So what is an "object", anyways? Three things are needed to completely specify an object: its position, its orientation, and its structure. The "structure" is the vertices and faces that define what the object looks like. The position and orientation determine where the object is, and in what direction it is pointing. Position and orientation are such important concepts that it is worthwhile to review them further.

Consider something like a chessboard. Each piece is located on a specific square, and when you move it, the piece moves to a different square (as opposed to e.g. moving the board). The square it sits on is its position, and in chess this is denoted by C-4 or something similar. Now imagine that we are one of the knights -- what would we see? It would depend on the direction we were facing; in other words, our orientation. If we were to then compute the view from a different knight, we would need to know its orientation.

In three dimensions, the position is specified by three signed 16-bit coordinates, the usual x/y/z Cartesian system. The orientation is specified by a 3x3 rotation matrix. This orientation matrix essentially tells you which direction is "forwards", which direction is "down", and which direction is "sideways", _from the object's perspective_. That is, if you're flying a plane, the "forwards" direction is always straight ahead. If you're watching it from the ground, the forwards direction changes if the plane turns, or dives, etc.

Thus, in obj3d, an object is defined by a 32-byte "object record":

Object:

CenterX	2 bytes	Position, x-coordinate
CenterY	2 bytes	Position, y-coordinate
CenterZ	2 bytes	Position, z-coordinate
Structure	2 bytes	Pointer to structure data
ID	1 byte	Optional ID byte
User byte	1 byte	Free data byte
CenterPos	1 byte	Position in rotated center list
CenXRem	1 byte	Remainder, x-position
CenYRem	1 byte	(Used by MoveUp, etc.)
CenZRem	1 byte	
Matrix	9 bytes	Viewpoint matrix, integer part
MatRem	9 bytes	Fractional part

CenterX/Y/Z are the 16-bit signed coordinates specifying where the object is located in the world. "Structure" is a pointer to the structure data defining the vertices and faces which make up the object. Using a pointer means that objects may share the same basic shape, without wasting lots of memory; it also means that objects may be e.g. animated (beyond merely rotating). When AddObj is called, the contents of .AY are stored here. The structure layout is discussed a little later.

The ID and user bytes are purely for the use of the programmer; they are *not* used by the library routines. For example, you might store what type of object this is, and its current velocity. When AddObj is called, the content of .X is stored in ID.

CenterPos is an index into a list of relative object centers; this list is generated when the viewpoint is calculated from a specific object.

CenX/Y/ZRem are used solely by the routines MoveForwards etc. They

represent the fractional portion of the center coordinates, to ensure accurate movement (especially when moving by small amounts).

Matrix and Matrem are the "viewpoint" matrix; the viewpoint matrix is the transpose of the orientation matrix. Values in Matrix range from -64 to 64. The remainder portion is used by the TurnLeft etc. routines (in particular, by the lib3d routines ACCROTX/Y/Z); only the integer portion is used for rotation/projection.

The viewpoint matrix determines what the world looks like when rotated about the object. The orientation matrix determines how the object looks from somewhere within the world, i.e. what direction it is pointing in, etc. It's the difference between being "inside" the object or "outside" of the object.

Computing views was explained in C=Hacking #16. Briefly, when an airplane e.g. rolls, it rolls about its fuselage -- about a local coordinate axis -- and when it rolls, the other coordinate axis change direction, relative to the world (e.g. the wings move). The problem is that the usual rotation matrix really only rotates about a _fixed_ coordinate system. If the object starts pointing in a different direction, you would need a way of rotating about the new coordinate axis, i.e. about an arbitrary line. It can be done, using quaternions (which are a generalization of complex numbers), but it is cumbersome and time-consuming (and thus popular among PC programmers). The smart way to do it is to realize that the inverse of a "viewpoint matrix" is an "orientation matrix", and to further realize that the inverse of any rotation matrix is simply its transpose.

By the way, it should go without saying that the object record values need to not get screwed up. If they do, all sorts of strange things can happen. So if you write a program, and it starts behaving or crashing mysteriously, first check that the object record isn't hosed.

The bytes following the user bytes will probably never be used by a programmer, except perhaps to copy orientation matrices. The bytes up through the user bytes, however, are meant to be modified by the programmer. For example, the next part of the example code sets the center of the object, after AddObj.

```
         LDA #<STARDAT
         LDY #>STARDAT
         LDX #02
         JSR AddObj
         STX ROB          ;Rotate object
         STA POINT        ;Object pointer
         STY POINT+1
         LDY #5           ;Set center
:l1      LDA OCEN,Y
         STA (POINT),Y
         DEY
         BPL :l1
```

Not only does AddObj return the object number in .X, it returns a pointer to the object in .AY. This pointer is also returned by the routines SetCurOb and GetCurOb. The above code stores this pointer in POINT, and then sets the location by copying from a little table

```
OCEN     DA 00            ;X-coord
         DA 00            ;Y-coord
         DA $0100         ;Z-coord
```

At the very least, you will probably have to specify the location of any object you create; otherwise, the default location is the center of the world, at (0,0,0).

Next, the example program tells obj3d about how objects should be rendered to the screen:

```
:sec     SEC              ;Solid polygons
:setp    LDX #$60         ;Bitmap at $6000
         LDA #<PATS       ;Pattern table
         LDY #>PATS
         JSR SetParms
```

SetParms takes four parameters. The carry flag determines whether solid or wireframe polygons will be used. (The wireframe routine is not particularly fast, because it is not particularly smart -- it does not compute hidden faces. It does, however, skip lines that have already been drawn.) .X contains the location of the bitmap (high byte) to be rendered to. .AY contains a pointer to a table of 8x8 bit patterns, used by the rendering routines:

```
PATS                    ;Pattern table
SOLID    = 0
         HEX FFFFFFFFFFFFFFFF
DITHER1  = 1
         HEX 55AA55AA55AA55AA
DITHER2  = 2
         HEX AA55AA55AA55AA55
         ...
```

These patterns will later be referenced by their position in the table.

Thus do we come to the main loop, which is just four routines:

```
:loop
         JSR ClrBitmap

         LDX VOB          ;Calculate view
         JSR CalcView
         JSR SortVis      ;Sort objects
         JSR DrawAllVis   ;Draw objects
```

The first routine is not a part of obj3d; it just clears the bitmap. JSR CalcView computes what the world looks like from object .X -- from the object's position and orientation. It does so by translating and rotating each object's center (except the viewpoint object), and storing this relative center in the center list; the value of CenterPos, in the object record, is an index into this list.

JSR SortVis computes a sorted list of _visible_ objects, from the center list. An object is considered "visible" if it lies in a 45 degree pyramid in front of the viewpoint object, and if its z-coordinate is between zmin and zmax. The paramters zmin and zmax may be set by calling JSR SetVisParms; I don't recall the default values offhand -- probably 100 < CenterZ < 8192. The purpose of the sorted list is to ensure the objects are drawn from back to front -- so that objects overlap each other correctly.

Finally, after calculating the viewpoint, and calculating the visible objects, all that remains is to draw all the visible objects, with JSR DrawAllVis. Individual objects may be drawn one at a time with JSR DrawNextVis. When an object is drawn, it is rotated by the orientation and viewpoint matrices, then added to its relative center (M R P + M C), then projected, with the projected points stored in two lists, PLISTX and PLISTY. The appropriate rendering routine is then called, using the points in these lists. Also, before projecting, the rotated z-coordinates are stored in PLISTZ, for use in drawing compound objects (described later).

That takes care of displaying an object. What about manipulating an object, i.e. moving around? The example program accepts the following keys:

```
a/z s/d q/w     -- Pitch, roll, and yaw object
@ /             -- Move forwards/backwards
space           -- Switch viewpoints
=               -- Draw using wireframe graphics
*               -- Draw using solid graphics
```

You can look at the source code to see how all of this is done. Here, it will be enough to look at the code to pitch the object and the code to move the object (and to swap viewpoints, for good measure). First, the library must be told which object to operate on:

```
         LDX ROB          ;Set rotation object
         JSR SetCurOb
```

Recall that after adding in the second object, .X was stored in ROB. In this program, movement always affects the second object, no matter which object is the view object. The program now waits for a keypress, and branches accordingly:

```
:wait    JSR GETIN
         BEQ :wait
         CMP #'a'
         BEQ :pitchdn
         CMP #'z'
         BEQ :pitchup
         CMP #'@'
         BEQ :movf
         CMP #'/'
         BEQ :movb
         CMP #' '
         BNE :wait
```

```
         LDX VOB          ;Swap viewpoint
         JSR SetCurOb
         JSR GetNextOb
         STX VOB
         JMP :loop

:pitchdn CLC
         DFB $24
:pitchup SEC
         JSR Pitch
         JMP :loop

:movf    LDA #$07
         DFB $2C
:movb    LDA #$F9
         JSR MoveForwards
         JMP :loop
```

To change the viewpoint, the program simply sets the current object to the current view object, then advances to the next object in the active object list, and stores that as the new viewpoint object.

The routines Pitch, Roll, and Yaw all function similarly. They all rotate by a fixed amount (2*pi/128 radians, which is about 3 degrees), and the carry flag specifies the direction of rotation. The rotation is performed about the _local_ coordinate axis of the object, which rotates with the object; there is also a routine, SetMat, which can rotate about the fixed world axis.

The MoveForwards, MoveDown, and MoveSide routines also function similarly. In these routines, .A contains the "velocity" or length of the move, as a signed 8-bit number; the actual distance moved is four times .A.

And that more or less sums up the less-technical aspects of the obj3d library routines. For a complete list of routines, routine parameters, and a memory map, see the files "obj3d.ref" and "lib3d.ref" in the .zip archive (they're plain ASCII text files). There's also some example source code. And now it is time for the more technical details: object structure data, compound objects, some internals, and lib3d v2.0 updates.

The More Techincal Details

And that more or less sums up the less-technical aspects of the obj3d library routines. For a complete list of routines, routine parameters, and a memory map, see the files "obj3d.ref" and "lib3d.ref" in the .zip archive (they're plain ASCII text files). There's also some example source code. And now it is time for the more technical details: object structure data, compound objects, some internals, and lib3d v2.0 updates.

As you may recall, the object record contains a pointer to the object structure data -- the data describing the points and faces that define what the object looks like. For a normal object, this data is organized as follows:

Structure:

TypeID = 0	1 byte	Object type (normal, compound)
Npoints	1 byte	Number of points (vertices)
Nfaces	1 byte	Number of faces
Xcoords	n bytes	Vertices, x-coordinates
Ycoords	n bytes	Vertices, y-coordinates
Zcoords	n bytes	Vertices, z-coordinates
faces:		List of faces
nverts	1 byte	Number of vertices in face
fillpat	1 byte	Fill pattern (index into pattern list)
fpoints	m+1 bytes	Points (indices into X/Y/Z above)

TypeID identifies the object type, and will be discussed in more detail shortly. Npoints and Nfaces do what they say; there is no limit on the number of faces, but an object may not have more than 128 points (which is a HUGE object!).

X/Y/Zcoords are signed integers in the range -95..95, for reasons described in C=Hacking #16. Note that the _length_ of each of these points must be less than 95. For example, a point like (65,65,65) has length sqrt(65^2 + 65^2 + 65^2) = 65*sqrt(3) = 112 or so, so if this point were rotated down onto the x-axis it would have coordinates (112,0,0), which is out of the -95..95 range. Strange things will happen to your rotating objects if you make this error, so if your objects start freaking out upon rotation, check this first.

Note, moreover, that the vertices are defined about the center of rotation of the object; this center is exactly the coordinates CenterX/Y/Z in the object record, i.e. where the object is located in the world.

Following the coordinates is a list of faces, which are specified by a fill pattern and a list of vertices. The fill pattern is an index into the pattern table, which you may recall is set using JSR SetParms; the actual location of the pattern is PATTAB + 8*fillpat. The vertex list is simply a list of indices into the X/Y/Zcoords table; moreover, this

list must close upon itself, e.g. 0-1-2-0, so that it is of length nverts+1, where nverts is the number vertices in the face.

With all that in mind, here is the data for the first object in the example program, a simple tetrahedron:

```
*
* Test object 1: simple tetrahedron
*
TETDAT
         DFB 0            ;Normal object
         DFB 4            ;Number of points
         DFB 4            ;Number of faces

* Point list

TETX     DFB 45,45,0-45,0-45
TETY     DFB 45,0-45,45,0-45
TETZ     DFB 45,0-45,0-45,45

* Face list

FACE1    DFB 3            ;Number of vertices
         DFB SOLID        ;Fill pattern
         DFB 0,1,2,0      ;Vertices

FACE2    DFB 3
         DFB ZIGS
         DFB 3,2,1,3

FACE3    DFB 3
         DFB CROSSSM
         DFB 3,0,2,3

FACE4    DFB 3
         DFB HOLES
         DFB 3,1,0,3
```

Pretty straightforward, really. Note that faces can have any number of points in them; triangles will have three points, squares (e.g. sides of cubes) will have four points, and so on. Thus the way to think about creating an object is that some points are defined about some center, and those points are then connected together to make faces.

This kind of definition works fine for simple objects -- that is, for convex objects, where one part of the object won't obscure a different part. But what about more complicated objects -- a tank, say, or a cross -- where one piece of the object might be in front of another piece at one orientation, and behind in a different orientation? The general problem of polygon clipping is fairly complicated and time consuming. Being 64 programmers, though, we are of course a little sneakier about doing stuff, and we can make suitably complicated objects by just modifying the above structure a little bit.

Consider for a moment a cross, or the pointy stars from Cool World. The basic problem is that a given chunk of the object might be either in front of the rest of the object or behind it. If it is behind, then we want to draw it first; if it's in front, then we want to draw it last. The idea, then, is to divide a complex object up into a number of smaller objects, and to draw them in the right order. In the obj3d lexicon, such an object is a _compound object_.

Compound Objects

To define a "normal" object, we define a list of points and then join points into faces. A compound object takes this a step further, joining faces into smaller "oblets". All that remains is to figure out what order to draw each oblet in.

Recall that before rendering objects to the screen, JSR SortVis is called. SortVis sorts the visible objects based on their z-coordinates and far-away objects are drawn first, so that objects will be rendered in front of one another on the screen. So we can do the exact same thing with the oblets: associate a _reference point_ with each oblet, depth-sort the reference points, and then draw the oblets from back to front. The reference point doesn't have to be connected to anything, or a part of any face -- just some point which allows the oblets to be sorted correctly.

As an example, consider the pointy stars from Cool World. These are easy to create, by starting with a cube, and then pulling out the centers of each face, creating a star with six tines. The tips of each tine are a convenient reference point, so the idea is to divide the star up into six oblets -- the tines -- and use the tip of the tine as the reference point for each oblet. Once the tips are sorted, the tines may be drawn in the correct order, and viola! Violin! Instant polygon

clipping, with very little computational effort. Note that the key to success with this method is choosing the reference points well.

The format for a compound object is similar to a normal object, with TypeID set to $80 and Nfaces replaced by Noblets:

```
Structure:
        TypeID = $80    1 byte          Object type (normal, compound)
        Npoints         1 byte          Number of points (vertices)
        Noblets         1 byte          Number of oblets
        Xcoords         n bytes         Vertices, x-coordinates
        Ycoords         n bytes         Vertices, y-coordinates
        Zcoords         n bytes         Vertices, z-coordinates

        oblets:                         List of oblets
          ref points    p bytes         List of reference points (indices)

          oblet 1
            nbytes      1 byte          Number of bytes in this oblet
            nfaces      1 byte          Number of faces

            face list:
                nverts  1 byte          Number of vertices in face
                fillpat 1 byte          Fill pattern (index into pattern list)
                fpoints m+1 bytes       Points (indices into X/Y/Z above)

          oblet 2
            ...
```

The list of reference points is a list of indices into X/Y/Zcoords, and the first point in the list is the reference point for the first oblet, etc. Note the "nbytes" field in each object; this is used to advance through the oblet list (for example, to find oblet #4 quickly). There is a limit of 32 or so oblets allowed; chances are awfully good you'll never reach this limit, consdering that an object can only have 128 points!

Here is the second object from the example program, which is just a modified pointy star. Note that the TypeID is set to $80; a normal object has TypeID = 0. It is also worth pointing out that the choice of reference points -- the tip of each surface -- can lead to incorrect clipping on occasion, because the tips are of different lengths; nevertheless, they work quite well.

```
*
* Test object 2: a compound object,
* spaceship-kind of thing (essentially
* cool world stars)
*
STARDAT

         DFB $80          ;Compound object
         DFB 14           ;Number of points
         DFB 6            ;Number of oblets

* Point list

         DFB 50,0-50,0,0,0,0,15,15,15,15,0-15,0-15,0-15,0-15
         DFB 0,0,16,0-26,0,0,10,10,0-10,0-10,10,10,0-10,0-10
         DFB 0,0,0,0,94,0-22,15,0-15,0-15,15,15,0-15,0-15,15

* Oblet list: reference points

         DFB 0            ;First 6 points
         DFB 1
         DFB 2
         DFB 3
         DFB 4
         DFB 5

* Oblet 1

         DFB 26           ;26 bytes
         DFB 4            ;4 faces

* faces
         DFB 3            ;4 points
         DFB SOLID        ;pattern
         DFB 0,8,7,0      ;Star 2, Tine 0, face 1

         DFB 3
         DFB ZIGS
         DFB 0,7,6,0
```

```
         DFB 3
         DFB ZAGS
         DFB 0,6,9,0

         DFB 3
         DFB DITHER1
         DFB 0,9,8,0

* Oblet 2

         DFB 26           ;26 bytes
         DFB 4            ;4 faces


         DFB 3            ;4 points
         DFB ZIGS
         DFB 1,11,12,1

         DFB 3
         DFB BRICK
         DFB 1,12,13,1

         DFB 3
         DFB DITHER2
         DFB 1,13,10,1

         DFB 3
         DFB ZAGS
         DFB 1,10,11,1

* Oblet 3
         ...
```

The remaining oblets are all similar.

lib3d v2.0

Finally, lib3d has been updated, to support the obj3d library and to be more cool in general.

The first thing to notice is that there are now two libraries: one for hires, and another for multicolor. The multicolor version not only renders in multicolor, but corrects for the screen aspect ratio by multiplying all y-coordinates by 5/4 after projection.

The second thing to notice is the addition of three new routines: PLOT, DRAWLINE, and VERSION. VERSION returns the lib3d version number; the high bit set indicates a multicolor version. Currently lib3d is at v2.0, so this routine returns either 2 or $82.

PLOT and DRAWLINE are used to, well, plot points and draw lines. The coordinates are 16-bit signed values, i.e. the routines understand points that are off the screen. To use them, you just store the coordinates in zero page and JSR. The line drawing routine is not the world's fastest, but it is still zippy, especially for its (quite small) size and flexibility.

The third thing to notice is that the memory map has been altered significantly, both in zero page and in RAM. lib3d now starts at $8400, instead of $8600, so not only has the jump table moved but there is no longer room for sprites in bank 2. Zero page has been shuffled around, with perhaps the most significant result being that all of the $Fx locations (everything above $C4 actually) are free for use by the programmer. See the memory map for more details.

What you might not have noticed is that several of the routines have now changed. The accumulation routines ACCROTX/Y/Z now require a pointer to the matrix to be accumulated; they used to simply operate on a matrix in zero page. This way, object matrices may be directly manipulated, instead of doing any wasteful copying to and from ZP.

ROTPROJ has also gone through a big change. Recall that the equation for a 3D world is

M R P + M C

where P is an object point, R is the local rotation (orientation), M is the viewpoint rotation, and C is the object's center. ROTPROJ used to just calculate MP + MC; it now calculates MRP + MC, which is why zero page now contains a viewpoint _and_ an orientation matrix. Also, ROTPROJ stores the z-coordinates before projecting, for use in drawing compound objects, so there's now a pointer to PLISTZ. Finally, I feel that rotation

without projection is now a pretty useless feature; I left it in, but it now stores the rotated points to PLISTX/Y/Z, instead of using a special zero-page pointer for the purpose.

There were also some miscellaneous recodings, to conserve memory and such, but they're nothing special. For descriptions of the individual routines and their use, see the lib3d.ref document in the .zip file.

The End

Since most of the obj3d routines are either really stupid or were yanked out of Cool World, I don't think there's any reason to go over them (except the cool sorting algorithm, discussed elsewhere in this issue).

What more is there to say?

Go home and code!

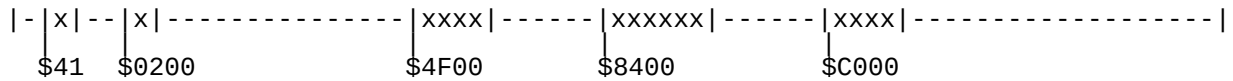
.....
....
..
.

::

Obj3d programmer's reference

last update: 4/16/99
version 2.0

* Obj3d and lib3d memory map:



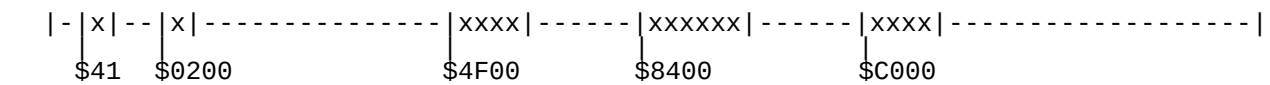
$41-$49	Orientation matrix
$4A-$52	Viewpoint matrix
$53,$54	Xoffset, Yoffset (screen offsets -- where 0,0 is on screen)
$55-$72	Various temporary pointers and variables
$60-$67	X1,Y1,X2,Y2 -- signed 16-bit coordinates for PLOT and DRAWLINE
$8B-$8E	PLISTZLO, PLISTZHI -- pointers to rotated z-coordaintes
$A3-$AE	CXLO, CXHI, CYLO, CYHI, CZLO, CZHI -- Pointers to rotated centers.
$AF-$B0	ROTMATH -- pointer to rotation math table
$B1-$B8	MULTLO1 LO2 HI1 HI2 -- pointers to multiplcation tables
$B9	Bitmap (high byte)
$BB-$BC	FILLPAT -- pointer to fill pattern (not table)
$BD-$C4	PLISTXLO, XHI, YLO, YHI -- pointers to rotated/projected points (PLISTZLO = $8B, above)
$0200	Point queue
$4F00	LINELO -- Record drawn lines in wireframe
$4F68	LINEHI
$4FD0	OBLETS -- Sorted oblet list
$5000	obj3d
$5600-$5D00	obj3d tables:
$5600	OBJLO -- Object list (pointers)
$5680	OBJHI
$5700	VISOBJ -- Visible object list
$5781	OBCEN -- Center object number
$5800	CX - Translated rotated centers
$5880	HCX - (high byte)
$5900	CY
$5980	HCY
$5A00	CZ
$5A80	HCZ
$5B00	PLISTX - Point list (projected)
$5B80	(high bytes)
$5C00	PLISTY
$5C80	(high byte PLISTY)
$5D00	PLISTZ
$5D80	high byte
$5E00-$5FFF	ROTMATH -- rotation table (relocatable, pointed to by $AF)
$8400-$9FFF	lib3d
$C000-$C2FF	MULTLO, pointed to by $F7-$F8 and $F9-$FA
$C300-$C5FF	MULTHI, pointed to by $FB-$FC and $FD-$FE

$C600-$CFFF   More tables (nonrelocatable, see lib3d.text)

So, to put it another way, free areas of RAM are:

$02-$40
$55-$72 (temporary only; library work variables)
$90-$A2
$C5-$FF

$02xx-$4FFF
$6000-$83FF
$A000-$BFFF
$D000-$FFFF



Moreover, the range $C000-$C5FF can be made available by relocating the
tables there. This way, $E000 may be used as a bitmap, with enough extra
room for 8 sprite definitions. Thus, bitmaps are available in all banks.


```
* Object record

ObjCX    = 0            ;Center, X-coord (signed 16-bit)
ObjCY    = 2            ;Y-coord
ObjCZ    = 4            ;z-coord
ObjData  = 6            ;Pointer to structure data
ObjID    = 8            ;User bytes
ObjUser  = 9
ObCenPos = 10           ;Position in center list
ObjCXRem = 11           ;Center remainders
ObjCYRem = 12
ObjCZRem = 13
ObjMat   = 14           ;Viewpoint matrix, int + rem
ObjSize  = 32           ;32 bytes total


* Jump table

Init3D      = $5000             ;Initialize lib3d
AddObj      = Init3D+3          ;Add object to object list
DelObj      = AddObj+3          ;Delete object from list
SetCurOb    = DelObj+3          ;Set current object
GetCurOb    = SetCurOb+3        ;Get current object
GetNextOb   = GetCurOb+3        ;Get next object in list
GetObj      = GetNextOb+3       ;Get pointer to object

SetMat      = GetObj+3          ;Calculate and set object matrix
Pitch       = SetMat+3          ;Pitch - rotate object around x-axis
Yaw         = Pitch+3           ;Yaw - rotate around y-axis
Roll        = Yaw+3             ;Roll - rotate around z-axis
MoveSide    = Roll+3            ;Move object
MoveUp      = MoveSide+3
MoveForwards = MoveUp+3
GetSideVec  = MoveForwards+3    ;Orientation vectors
GetUpVec    = GetSideVec+3      ;(length=64)
GetFrontVec = GetUpVec+3

SetParms    = GetFrontVec+3     ;Set rendering parameters

SetVisParms = SetParms+3        ;Set visibility parameters
CalcView    = SetVisParms+3     ;Set viewpoint = object
SortVis     = CalcView+3        ;Compute and sort visible objects
DrawAllVis  = SortVis+3         ;Draw all visible objects
GetNextVis  = DrawAllVis+3      ;Draw next objesible object list
RotDraw     = GetNextVis+3      ;Rotate and draw object
DrawFace    = RotDraw+3         ;Draw single face (polygon)

* lib3d stuff

CALCMAT  EQU $8800
ACCROTX  EQU $8803
ACCROTY  EQU $8806
ACCROTZ  EQU $8809
GLOBROT  EQU $880C
ROTPROJ  EQU $880F
POLYFILL EQU $8812
```

```
PLOT     EQU $8815
DRAWLINE EQU $8818
VERSION  EQU $881B


Routine descriptions
--------------------

Init3D
  On entry: .AY = pointer to object record storage area
  On exit:

  This routine is used to initialize various obj3d and lib3d pointers and
  variables, and should be called before any other routines are called.
  Note that if e.g. tables or the screen center are moved around, the
  corresponding variables should be set *after* calling Init3D.

AddObj
  On entry: .AY = pointer to object structure data
            .X  = optional user ID byte
  On exit:  .AY = pointer to object
            .X  = object number
            C set indicates error (e.g. too many objects!)

  AddObj is used to add objects into the world.  It places the object at
  the first empty spot in the active object list, and allocates a
  corresponding portion of memory in the object storage area, as passed
  to Init3D.

SetCurOb
  On entry: .X  = object number
  On exit:  .AY = pointer to object
            .X  = object number

  SetCurOb is used to set the current object.  The "object number" is
  a number returned by AddObj.  This routine is used before calling
  e.g. the movement routines, which act on the current object.

GetCurOb
  On entry:
  On exit:  .AY = pointer to object
            .X  = object number

  GetCurOb is used to get the current object number and pointer.

GetOb
  On entry: .X  = object number
  On exit:  .AY = pointer to object

  GetOb gets a pointer to an object without setting that object to
  be the current object.

DelObj
  On entry: .X  = object number
  On exit:  C set means rather nasty error

  DelObj is used to delete an object from the active object list.

GetNextOb
  On entry:
  On exit: .X  = object number
           .AY = object pointer
           C = 1 -> error

  GetNextOb gets the next object in the active object list, starting from
  the current object.  On exit, the current object is set to .X.  This
  routine may be used to cycle through the list of active objects.


* Object manipulation routines


SetMat
  On entry: .X  = angle around x-axis
            .Y  = angle around y-axis
            .A  = angle around z-axis
  On exit:

  SetMat is used to set a rotation matrix for the current object.  Angles
  go from 0..127.  Note that SetMat rotates around the fixed _world_ axis,
  not an object's local coordinate axis; use Yaw/Pitch/Roll to rotate about
```

the local axis.

Yaw
Pitch
Roll
  On entry: C clear -> positive rotation
            C set   -> negative rotation
  On exit:

  Yaw, Pitch, and Roll rotate an object by a fixed amount (3 degrees or so)
  about the local coordinate axis. (The local coordinate axis rotates
  with the object).

MoveUp
MoveSide
MoveForwards
  On entry: .A = distance (signed)
  On exit:

  MoveUp/Side/Forwards are used to move an object along its local coordinate
  axis, by an amount proportional to .A; negative values of .A will move in
  the opposite direction.

GetUpVec
GetSideVec
GetFrontVec
  On entry:
  On exit: (.X,.Y,.A) = signed (x,y,z) coordinates of orientation vector

  GetVec is used to figure out what direction the current object is
  pointing in. The vectors are signed, and of length 64.

* Visualization routines

SetParms
  On entry: .AY = pointer to pattern table
            .X = Bitmap address (high byte)
            C set   -> solid polygons
            C clear -> wireframe
  On exit:

  SetParms is used to set certain rendering parameters. The pattern table
  consists of a list of 8x8 patterns.

SetVisParms
  On entry: .AY = Maximum object range
            .X = Minimum object range
  On exit:

  SetVisParms is used to set the range in which an object is considered
  "visible". An object's center z-coordinate is compared to these values;
  the range is not a radius.

CalcView
  On entry: .X = Viewpoint object
  On exit:

  CalcView computes the view from the viewpoint object, by translating
  and rotating centers relative to the viewpoint object. The relative
  centers are stored in the CX HCX etc. lists; the ObCenPos element in
  the object record is an index into this list.

SortVis
  On entry:
  On exit:

  SortVis computes a sorted list of visible objects. Objects are visible
  if they are within a 45 degree cone forwards of the viewpoint object,
  and within the min/max visibility range, which may be changed with
  SetVisParms. CalcView must be called before calling this routine.
  Sorting is necessary to ensure that objects overlap one another
  correctly on the screen. On exit, the current object is set to the
  first object in this list.

DrawAllVis
  On entry:
  On exit:

  DrawAllVis draws all visible objects, in order, as determined by SortVis.

```
GetNextVis
  On entry:
  On exit:  .X  = Current object
            .AY = Pointer to object
            N set -> at end of list

  GetNextVis fetches the next object in the visible object list, setting
  it to the current object.  This routine is used to draw one object at
  a time, but in the proper order.

RotDraw
  On entry: .X = Object number
  On exit: Cool rendered polygon

  RotDraw renders an object to the screen, by rotating and projecting
  the object vertices and then drawing faces, either wireframe or solid.
  On exit, the rotated and projected points, i.e. the screen coordinates
  of the object, are stored in PLISTX and PLISTY; PLISTZ contains the
  rotated (but not projected) z-coordinates.

DrawFace
  On entry: .AY = pointer to face data
  On exit:  .AY = pointer to next face

  DrawFace renders a polygon to the screen, either wireframe or solid.
  It would be an awfully good idea to have the rotated points in PLIST,
  i.e. to call RotDraw before calling this routine.
.......
....
..
.                                   C=H #18

::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::


Stroids    o
-------  o   o
              o SLJ

        When you get down to it, just displaying things on the screen is
awfully easy.  And clearly, the obj3d library is up to the task of simple
programs like Cool World.  What isn't clear, however, is whether it is
useful for making applications like games, where objects are constantly
being created and destroyed, and have different properties and velocities
and such.

        Hence stroids.  Stroids is a demonstration of the obj3d routines,
more specifically a demonstration that obj3d is suitable for writing games
and such.  In stroids, you can fly around in an asteroid field containing
three different types of randomly drifting asteroids.  The first part of
this article will discuss the main algorithms and techniques used in the
program.

        Stroids is also meant to be a starter project for people who would
like to experiment with the library.  For the most part, all of the difficult
routines have been written; with a little more work, Stroids could be made
into an actual game, and there are lots of opportunities for modification and
improvement.  Thus the second part of this article will suggest a number
of possible ideas and modifications, and how they might be coded up.  This is
rather akin to a C=Hacking C=Hallenge, so I'm interested in publishing the
niftier modifications and applications that might result.

        Finally, the full source code is included at the end of this article
(the PETSCII el cheapo version is in the .zip).


The program
-----------

        The source code is in El Cheapo format.  To use the program, just
load and run the loader program.  Stroids accepts the following keys:

        joystick in port 2      Pitch and yaw, fire
        @ and /                 Increment/decrement velocity
        space                   Switch viewpoint object

        r                       Toggle radar on/off
        a/z q/w s/d             Pitch, roll, yaw
        +/-                     Increment/decrement asteroid density
        =                       Toggle wireframe mode
```

Initially, the viewpoint is set to a large, fixed tetrahedron at the center of the world; your ship is straight ahead, and you can move it around using the joystick and @/. Pressing space shifts the viewpoint to the next object -- the ship. Pressing space again will hop onto the next object -- an asteroid! Pressing it some more will cycle through all the objects, eventually returning to the first object.

The code

The purpose of this project was to test out the obj3d routines in some tougher game-like conditions, in which many non-indentical objects are present and are being created and destroyed more or less randomly, and where things like collisions must be detectable. A 3D asteroid field seemed like a reasonable test -- asteroids can created more or less randomly, with different sizes and velocities. A player could then fly around in the asteroid field, maybe with a radar display showing the relative positions of asteroids. Maybe the asteroids could be shot, or simply dodged. And so on.

The program is really pretty simple. There are two routines which control the asteroid field; one controls creating and destroying asteroids, the other controls their movement. There are two routines to compute and render a crude radar display. Obj3d is used to render the screen, and all that's left is to move the ship around and wait for keys. A little IRQ routine is used to keep the program running under 60fps on a SuperCPU, which also reads the joystick (and which in principle would play music, handle sound effects, etc.), and that's about it. Thus the main loop is quite straightforward:

- Update asteroid field
- Set up radar
- Compute/render main screen
- Render radar display
- Move ship and wait for keys

```
Stroids
         JSR Init
         LDA #00
         STA CURD

:loop
         JSR CheckDensity
         JSR DriftStroids

         JSR SetRadar

         LDX VOB          ;Calculate view
         JSR CalcView
         JSR SortVis      ;Sort objects
         JSR DrawAllVis   ;Draw objects

         JSR DrawRadar

         JSR CheckFire
         JSR SwapBuf

         LDX ROB          ;Set rotation object
         JSR SetCurOb
         LDA VELOCITY
         JSR MoveForwards

:wait    LDA IRQFLAG
         BEQ :wait
```

The asteroid field

The asteroid field only "exists" in a little cube. When asteroids exit this cube they are destroyed, and new asteroids created. Currently this cube stays fixed in the world, but one of the suggested projects is to center it on, i.e. make it move with, the ship. The purpose of the cube is to keep things manageable -- it makes no sense to keep track of asteroids that are many thousands of units away. That would just massively bog down the system, and mean that the player wouldn't ever see or interact with any asteroids. What's fun about that?

The asteroid field within the cube is characterized by a "density", which may be changed by pressing the + and - keys. There are three different kinds of asteroids -- large, medium, and small -- and each asteroid has a

"weight". When an asteroid is created, its weight is added to the total field weight; when destroyed, its weight is subtracted. If the total weight exceeds the "density" value, no new asteroids are created. The net result is a constantly changing asteroid field with a pleasantly uniform density.

To create an asteroid, a random value is chosen for the weight; this weight determines the size of the asteroid. If the weight is too large (would exceed the field density) then no asteroid is created. This keeps the field changing, since asteroids are not simply replaced by identical objects, and probably gives big asteroids a chance. That is, a field that is always "full" surely favors small objects: if a large asteroid leaves the field, it can be replaced by multiple smaller ones; if a small asteroid leaves, it can only be replaced by a smaller one. So I predict that the field would otherwise quickly degenerate into a bunch of little asteroids.

If an asteroid makes it into the field, it is assigned a random position, speed, and orientation. The orientation is computed using the obj3d routine SetMat. The object's "weight" is stored in the ID byte of the object, and the velocity is stored in the user byte. This provides an easy way to identify the type of object, and keeps the velocity along with the object (instead of putting it in another table). By storing all non-asteroid objects with the high bit of the ID byte set, it is very easy to differentiate asteroids from other objects. Once created, an asteroid just moves in a straight line, until it leaves the play area. There is no collision detection -- that's a project for you! (But it's pretty easy).

All of this is done using two routines: CheckDensity and DriftStroids. CheckDensity simply creates new asteroids until it can't create any more; DriftStroids simply goes through the object list, looking for asteroids and moving any it finds (recall that the velocity of the asteroid is stored in the user byte). The code is pretty straightforward, so you can go through the source listing to examine the routines in more detail.

Random numbers

It is worthwhile to say a word or two about the random number generator used. I used it as an experiment in my quest for a very quick but reasonable random number generator, and it seems to work fairly well -- certainly better than other ones I tried. (Deficiencies in random number generators become awfully apparent awfully quick, in the asteroid field).

This particular generator uses the "middle-squares" method. In this method (which was proposed by VonNeumann), an n-bit number is squared, and the middle n-bits are taken as the next number in the sequence. In this case, the numbers are 8-bits. To get the next number in the sequence, the number is squared, giving a 16-bit number, and the middle eight bits -- bits 4 through 11 -- are used as the next number in the sequence.

To do the squaring, I simply used the lib3d multiplication tables of f(x)=x^2/4. Two more divisions by two thus gives x^2/16, i.e. the middle 8-bits.

Middle-squares is not without problems. The method is known to have a number of short sequences -- ideally you want a sequence of numbers that takes a long time to repeat; a sequence like 1 99 63 1 99 63 ... isn't so useful. The method also dies once you hit a zero -- zero squared is just zero. So I built in a simple modification to compensate for these deficiencies: the routine re-seeds itself whenever a) the current seed generates itself (like zero will), or b) the current number sequence exceeds a maximum length (that is, after generating N numbers it re-seeds itself). The re-seeding is just a simple method to keep the algorithm from getting stuck.

Radar Display

Navigating around a 3D world is very difficult without some form of reference. A compass is one possibility; landmarks another. A radar display is potentially much more useful, though, and also more difficult -- another good test of the routines. The stroids radar uses a pretty neat trick: it actually creates a "radar object", whose vertices are the object centers, and lets obj3d do the hard calculations. This will be explained shortly.

The purpose of the radar is to display where objects are located relative to the person. With a few moments of thought this is seen to be very similar to the "viewpoint" problem. That is, to compute the viewpoint from a particular object, the surrounding objects need to be translated and rotated about the viewpoint object. This is exactly what the obj3d routine CALCVIEW does. So by dipping into the obj3d center list (after calling CALCVIEW) we can get all the relative locations automatically.

The purpose, however, is to _display_ the relative object locations, not merely to compute them. Imagine, for a moment, a cube, with us located at the center of the cube (this is, after all, all a 3d world really is).

Then imagine a bunch of dots around the cube, representing the locations of other objects. There's our radar display -- now how could we draw something like that on the screen?

Well, we said it was contained in a cube. But drawing a cube is pretty easy -- just specify the vertices of the cube, locate the center out away from the viewer, rotate and project. Well from there it's awfully easy to compute the points inside the cube -- just add them to the list of vertices, and rotate and project them along with the rest of the cube, and then maybe plot them on the screen. And when you get down to it, the vertices of the cube aren't really necessary.

The only other thing to realize is that there is an obj3d routine for rotating and projecting points -- RotDraw. So now we have a way of generating a radar display: create a new object, whose vertices are exactly the relative object centers computed by CalcView. Give the object a center coordinate out a little ways from the viewpoint object, and then just rotate, project, and plot the resulting points somewhere on the screen.

The radar code is a little tricky, and could surely use some modification, so it's worth going through it in some detail. The radar is an actual obj3d object, and is the first object created -- thus it is object number 0. Recall that CalcView translates and rotates all objects relative to the viewpoint object. Before CalcView is called, the location of the radar object is set to the location of the viewpoint object:

```
*
* DrawRadar -- Compute and draw radar
* display, by creating a new object whose points are simply the
* (scaled) object centers.
*

RADOFF   = 24             ;Size of radar
RADFLAG  DFB #$00         ;Radar toggle

IDENTITY DFB 64,0,0       ;Identity matrix
         DFB 0,64,0
         DFB 0,0,64

* Set radar center to view center

SetRadar
         LDX VOB
         JSR SetCurOb
         STA TEMP
         STY TEMP+1
         LDX #00
         JSR SetCurOb
         STA POINT
         STY POINT+1
         LDY #5
:l       LDA (TEMP),Y
         STA (POINT),Y
         DEY
         BPL :l
rts2     RTS
```

Thus CalcView will translate the center to be (0,0,0) and store those zeros in the center list (CZ and HCZ, located at $5A00 and $5A80). One consequence of doing this is that the radar object will not be considered "visible", and hence skipped over by DrawAllVis -- the radar object needs to be drawn manually.

That's the job of DrawRadar. The radar is drawn on the screen last, so it is always in the foreground. The first step is to locate the radar as if it were out in front of the viewpoint object (if you think of the radar display as a cube, you can see that it needs to be treated as if it were out in front of the viewer, not right on top of/surrounding the viewer):

```
DrawRadar
         LDA RADFLAG
         BMI rts2
         LDX #00
         JSR GetObj
         STA POINT
         STY POINT+1

         LDY #ObCenPos
         LDA (POINT),Y
         TAX
         LDA #200         ;First, cheat the center
         STA CZ,X         ;by putting it at (0,0,x); note radar is object 0
         LDA #03          ;(so projection will work out correctly -- don't want
```

```
         STA HCZ,X        ;negative points!)
```

A value of $03C8 is stored directly into the object center list CZ. This is the list used by RotDraw: the object centers from this list are added to the object vertices before projecting. The value $03C8 was chosen because it makes the radar a reasonable size on the screen -- smaller values of the z-coordinate mean the object is closer to the viewer, and hence will appear larger.

The next step is to set the viewpoint orientation matrix to the identity matrix. RotDraw implements the equation M*(RP + C) -- that is, rotate a vertex P by the orientation matrix R, add it to the center C, and rotate the whole thing by the viewpoint orientation matrix M. In this case, the radar needs to be _not_ rotated, so M needs to be set to 1 (R, the local rotation matrix, is already set to the identity matrix). The viewpoint matrix used by RotDraw is stored in zero page, at VIEWMAT, so it's a simple matter of copying the identity matrix defined up above:

```
         LDX #8
:l1      LDA IDENTITY,X
         STA VIEWMAT,X    ;Don't add any extra viewpoint rotation
         DEX
         BPL :l1
```

The next step is to create the data for the radar.

The idea is to create an actual radar object, and update the vertices at each pass of the main loop. The radar object is stored at the very end of the code:

```
*
* Radar object.  To do the radar, an object is used whose points are simply
* the translated and rotated object centers.  This object is then projected
* using the obj3d routines, and the points plotted.
*
RADOBJ
         DFB 0            ;Normal object
         DFB 0            ;Number of points
         DFB 0            ;Number of faces

* Point list

RADDAT
```

The idea is to update the number of points and set the vertices -- stored at RADDAT -- to be the translated and rotated object centers. Recall, however, that object vertices must be in the range -95..95 or so. A typical object center, however, is a 16-bit number. So the object centers need to be scaled; the radar code divides them by eight, before adding to the vertex list, setting any out of range numbers to either -95 or 95. This is done using the little subroutine DIV8X1, used later.

Before doing anything, however, three pointers need to be set up which point to the x-, y-, and z-coordinates of the radar object vertices:

```
         LDX NUMOBJS
         DEX
         STX RADOBJ+1     ;Number of points
         TXA
         CLC
         ADC #<RADDAT
         STA TEMP         ;y-coords
         LDA #>RADDAT
         ADC #00
         STA TEMP+1
         TXA
         ADC TEMP
         STA POINT        ;z-coords
         LDA TEMP+1
         ADC #00
         STA POINT+1
```

Note the STX RADOBJ+1, which sets the number of points stored in the object. Once this is done, the code simply moves through the translated/rotated center list, divides each coordinate by eight, and stores the result in the object:

```
         LDY #00          ;Order doesn't matter
:loop
         LDA CX,X
         STA X1
         LDA HCX,X
         JSR DIV8X1
```

```
         STA RADDAT,Y

         LDA CY,X
         STA X1
         LDA HCY,X
         JSR DIV8X1
         STA (TEMP),Y

         LDA CZ,X
         STA X1
         LDA HCZ,X
         STA TAB1,Y       ;Remember sign
         JSR DIV8X1
         STA (POINT),Y
         INY
:skip    DEX
         BNE :loop        ;0 is radar object
```

That's all there is to creating the radar! But it still needs to be rendered to the screen.

The lib3d routines have two zero page variables, XOFFSET and YOFFSET, which tell the routines where the center of the screen is -- after points are projected, they are added to XOFFSET and YOFFSET to get the actual screen coordinates. Normally these are set to the physical center of the screen, e.g. XOFFSET=160 and YOFFSET=100, but having the radar plopped right into the middle of the screen isn't so useful -- better to have it tucked away off in a corner. The easiest way to do this is to change XOFFSET and YOFFSET to make the projection routines think that the origin is at, say, (24,24), in the upper-left corner of the screen:

```
         LDA XOFFSET      ;Change coordinate offsets
         PHA
         LDA YOFFSET
         PHA
         LDA #RADOFF
         STA XOFFSET      ;Upper-left cornerof screen
         STA YOFFSET

         LDX #00
         JSR RotDraw

         PLA
         STA YOFFSET
         PLA
         STA XOFFSET
```

RotDraw projects the points, and stores them in PLISTX and PLISTY. Therefore all that remains is to move through these lists and plot each point! When I tried this, however, the result was unsatisfactory -- there were just a bunch of points on the screen, and it was very difficult to tell where they were located relative to the view, or how far away they were, etc. So the next logical step was to draw lines instead of points -- that is, draw a line from the center of the radar (the viewer location) to the individual points. This helped, but it was still difficult to determine e.g. whether an asteroid was in front of or behind the viewer. So the last step was to use different colors for the lines, depending on whether the object is in front or behind the viewer. This is done by simply checking the sign of the z-coordinates; the z-coordinates were stored in a spare table TAB1, above, in the code which computed the object vertices.

With all that in mind, here's the rendering code:

```
:loop2   LDY RADOBJ+1     ;Now plot each point
         DEY
         LDA PLISTX,Y
         STA X1
         LDA PLISTX+$80,Y
         STA X1+1
         LDA PLISTY,Y
         STA Y1
         LDA PLISTY+$80,Y
         STA Y1+1
         LDA #RADOFF      ;Line from center to point
         STA X2
         STA Y2
         LDA #00
         STA X2+1
         STA Y2+1

         LDA #<COLOR2     ;Set fill pattern
         STA FILLPAT
```

```
         LDA #>COLOR2
         STA FILLPAT+1
         LDA TAB1,Y       ;Use different colors
         BMI :draw        ;for front/back
         LDA #<COLOR1
         STA FILLPAT
         LDA #>COLOR1
         STA FILLPAT+1

:draw    JSR DrawLine

         DEC RADOBJ+1
         BNE :loop2
:rts     RTS              ;Whew!
```

where COLOR1 and COLOR2 are simple 8x8 patterns:

```
COLOR1   HEX 5555555555555555
COLOR2   HEX AAAAAAAAAAAAAAAA
```

And that's all there is to it!
That pretty much wraps up the non-obvious parts of the code; the rest is really straightforward. All that leaves is...

Simple Modification and Improvement Projects

Not only did I not feel like doing further work on the project, it occured to me that others might actually be interested in doing the simple routines needed to e.g. make it into a game. So in this section I thought I'd describe some possible directions that the code might be taken in -- some simple projects that give some practice in using the obj3d routines and 3d graphics in general. So without further ado...

- Third person perspective

It might be interesting to try putting in a third-person perspective, i.e. place an empty object above and behind the view object.

- Collisions

Currently there is no collision detection -- in particular, there is no detection of an asteroid hitting the ship. This is a pretty easy task -- get the ship's center x-, y-, and z-coordinates, and compare them to the coordinates of every asteroid. Use JSR GetNextObj to traverse the object list, check the ID byte (asteroids have the high bit clear), and if an asteroid compute the difference between the asteroid and ship centers. If it is small enough for all three coordinates, signal a collision. Note that the translated/rotated centers in CX, CY, CZ may be used without performing the subtraction.
If you're feeling more brave, a good challenge would be to let asteroids collide, and break into smaller asteroids upon collision. It just involves many more comparisons -- perhaps even a little data structure to process the comparisons more efficiently (e.g. sorting one of the coordinates, and only comparing near neighbors).

- Realistic flight model

Currently stroids uses a rather cheesy "flight model", where you always move in the forwards direction. A more realistic flight model, taking intertia into account, would probably be more fun. Physics says that force causes a change in velocity in the direction of that force -- in other words, if you thrust in a particular direction, it adds velocity in that direction. To implement this on a computer, you need to get the direction of thrust, using GetFrontVec, and add that vector to the total velocity vector:

If thrust
- GetFrontVec (vector has length = 64)
- Scale vector -- maybe divide by 16, keeping fractional part; maybe treat strictly as fractional part.
- Add to total velocity

Then at each pass through the main loop the code simply adds the total velocity to the center coordinates.

- Compass, position, fuel, stats

With a more realistic flight model, it becomes imperative to somehow indicate what direction the ship is moving, compared with the

direction it is pointing. In the absense of landmarks, this means that the position and direction need to be indicated somehow, either graphically or in plain text. A simple solution would be to use a raster interrupt at the bottom of the screen, and display the information as text.

Naturally, if a game of some sort is desired, then it might be helpful to add damage and fuel into the program.

- Better radar

The current radar display is not terribly effective. One possibility is to make an Elite-like radar display, which gives a plane of reference and indicates altitude and position (i.e. draws right-triangles). This isn't a very difficult extension of the radar calculation. The radar display should be tilted, which means using SetMat to rotate the radar around the x-axis when it is first initialized. Then, when adding the object vertices, for every point (x, y, z) add another point (x, 0, z) -- drawing a line between these two points gives the altitude. (Literally, it draws the altitude of a right triangle; the current display draws the hypoteneuse).

- Better asteroid objects

Tetrahedrons are not particularly inspiring asteroids. Draw some new non-lame ones!

- Move asteroid field with ship

Currently the asteroid field is centered at (0,0,0) -- that is, the boundaries of the field are fixed. Center it instead on the ship, so that the boundaries move with the ship.

- Variable field density

Instead of having a single fixed density to the field, modify the program so that different areas of space have different asteroid densities.

- Projectiles and lasers

What's a game without weapons? A boring game, that's what. One possibility is to add a projectile weapon -- make a little cube shoot out the front of the ship, for example. It might fly straight, or it might home in on a target. A collision detection routine would also be needed.

Another thing to do is to make the lasers actually do something. Detecting a laser hit on an object is non-trivial -- ships which are closer are larger, and should be easier to hit. The basic issue is to determine whether or not the object covers the center of the screen -- that is, whether there is any point (0,0,z) _inside_ of the object. One way to do this is to check whether the (rotated and translated) center of a visible object is within a certain distance of (0,0,z) (i.e. only check the x- and y-coordinates). Note that CalcVis calculates and sorts the visible objects.

Another possibility is to draw each object, one at a time, and check to see if the center of the screen (or some small area of the screen) was plotted to. The thing that makes this method a little tricky is that the farthest objects are drawn first, so a program needs to keep track of the _last_ object that draws to the center of the screen, not the first. And doing that means clearing the pixels being checked whenever they are rendered to.

There are probably other, better methods of detecting collisions!

- Make a game

Naturally. Go crazy!

Source code

See the .zip file, or the Fridge, for the stroids source code.

....
....
..
.

C=H #18

::


```
begin 644 obj3d.zip
M4$L#!!0````(`#1CQQH3MMYJU`0```$&```'`!``;V)J,V0N;U58#`!\C'XW
M)(@3+)P/9`!=5&V,$T48GFN%G'`<"PHVDNC&'_B1,U?X52*)\L-/!KH?W6_.
MK,(!58_$F)CP@V1).F-3NTU+0M(#+FZ6[GE[WC57$,\JQ0.1E"JD)B1*C!&%
MGE5/TDM(!(QWSO1`C4WGG7?>9]ZO9V86<##&P6\Y^!P/-_!P(P]W\O!I'K[#
MPU,\/,O#&@\O\?!K'G[#P[<%^)8`]PKPB`!'!'A,@$^*\(P(OQ3AM`C72X#^
MV!V@TQ:XM,CY+Z,)]S0^5L,?NF?Q1S7\L3]Q;^%\8&_F[*T<JJ#3Z%W?0I^A
MSY'M&O@4GFKB, AL$Z=QQ@7X*'X??^!:V,,^'G=57"!CV-7P"!FCKH['R"BZ
```

M[SG;V35))N!&L9B(.2`_:,E5YK:)I(2"91>DD!HZ:<FMKCB;#4ZKH28S&S$'
M!`[I0T#VU`&10\:0)5_?S,ULY;+!S%;.>=@'.3W)_.D_XW3G=&<E&:O)Z/3D
MG.XI22(D(C);N.(6K@)D$MWT`8DQOIEKK;JX$&A9Q9+KBVK,S4B3A#0I]"#9
M'EY%P8`[WZ0UW2;>)$S(C)@L2%+X$6()Q;N13MSC`!D^<%:06J8ZZ[?*F]I5
MW30[V8T\V9/MJ#YFLL=X&$Q2N;@M[TV:3G>/LZ+'N3^AG`>M651")UZ<#\4[
MT'$T23/\0#*TPR.CI(_X2UW`/B04E&I"H<LN-["PI(ME[CU81E*I%)D]CM32
MB=G)*2L;:)2&2\RBA!9I:$ASP./S3$<ROUA=JSWQ/S'JJ04I]*:>TPNRI]&Y
MFJ+RGPII.0_\MYS1*LG[0I+DWFV2J\3L&13L0T+ZL(#+IAWETAR7X3DSDQ?8
M#3S2$X;;Z:PNZ0=?2A*>W:4GCA[<7F-ND9LCLSOYF;S0>AZI"2TU+`_M5U+.
MDISJ@)(:F24'KU9+*IDM.K<-VAV#=M>@WS'HTS)3*LAIR,&N)"!WMT?T+=M2
MBI"KA?=6+!UI%:`C]4*4\[3?.6ZBSU,O\$3_!60%3PWM`ZJG[;/4\+.>^CW5
M_R+ZAC:@44`+]U!`HX`69AVK#)3PRH0R4@;Z%VK9TAO:Q$R^H`P!)34(E!J3
M=JSTI&@"(LK[E5%V6BQ.BB>!PEPW%Y3P'P0ZM5\AM<Y)Y%&W7^6_+#O=+LC,
M25CU@T@CY&VJ-IO-<UK]QK#:O!*L_^PL*>E(<A:7=/M'<92<A_V3B`RL$^YT
M^YI(#"W#ODIGNR%6\Z%^'<6'C6R@&<*O#5P5T>MM]0VJ#K35/42UPU(J#NQU
MTMA5<5B*L,N3XPVQ==]X6!I;)Q6OB>$N=KUTL2'6;])SIT<;&FB(=BZ6)UX'
M8GZO/1\E\%A#))>D`OJ06K'ZD%9LB$YON1X]WSNQ0%D?I:R/4/8KI:P>30W6
MJ1]SKKUM-).-7>F=>&K\0`P9X[D8TJ^`UG('D#N*=/)Q,$*U^@VGH]U9P?#T
M4#Q`'MG=RL:R,>:3N^6UV>B5(O%`OTZ:\P%2^PW2'_5>LE9=^`\<$M"GY!-S
M6$`GBX,"LX8\>\+;$`C,]$KU&ZE6D%V97.#`=`+MS@QD5BP#O5(!V]"K%6L;
MVHZEZAUD!T7Z*;*3(KL2B@.^FI-:3.HRB-8?]:3+NZ.MU57F-T\:!*3UW=&I
M7:W@]3F)#24+RD62\DAD(DLVU;\C>XG+I583_`U02P,$%`````@`-F/'&DNF
M#.+S"P`Z!L```D`$`!L:6(S9'8R+F]56`P`@8Q^-R>($RR<#V0`Y5A]<!/7
M$3_;^)L/A0`Y"($K0QA("54SF8RFS8=M+",[9TQL2_(7]NFDDT[^E)1.!]HF
M\1]ZLF.P+8%-1(#$X]Q3+"5I=(5,P@379A@<VTF+_F@ZR;29H3.1!]I,*C+Y
M`](F=-_I3.PB#"'DK^QH[??V[>Y[N[^]/3U17DJE-)725<I0:9%*F2IEJ92M
M4HY*N2KEJ92OTF*5EJBT5*5E*FE4NDNEY2K=K=(*E5:JM$JE>U2B55JMTAJ5
M[E5IK4KWJ;1.I?4J49WCYQ.7__O-C_7_<T^L7Y:=D4;]6/\/48&VH1Q@#3`-
MS`!O!M8"ZX`+@`W`NX!K@#E@$=@%O`>X$[@;V`\<!!X"'@&.`K\#/`X\"1P#
M_@CX//`%X`3PY4#;3/O-3L"Q5!=5T#F>DJFT](S,K.R<O/S%2Y9I[EJ^8N6J
M>U:ON7?M.BI'0S.;M;H"PZX:3G3MZ>SV!X=&HN^,3\8^.G\A<7EVG=WP(OOP
M$=9UA/WI$3;_*!LXQN[O9O5!]IE@.)VC%N4LUJR@US(;-V_5/JQ[M*#8P.ZJ
MKFG@;&*KZU=[?M?IZ^D[<.C%EX;3L_*6+E^U9MV&30]L>^B17SQ>5%*VL])4
MU\C;F]L]O_[-L][N_?Z!X-$A2Z6>J:TP,M4&/?-DZ<X=3$6),B[3FZN>\-7X
M:L,4,B(3JCM%I<NGSQKE,]JE<9,_SU<GGT8P0Z;IV*>ANE@:YZLG*J/(AASR
M&!*0B(TEIN;D![5$L'%+-FJ^+]2%VD;'NFCA)&I'':&ZB5I-[DP#KIVZ&AWE
MF.%#VJ\R&Q+_]-5)1N*$N)-,NDNCR(6%2V.:##:]![F9,X<8__.221Z+R*.2
M45GN45;[815LR`%$9L,+R(1MNKB#&,=%Y,8N79R+9DRX=7$>9.Z+E#8STY5(
MXZ26\=713&QB*_:/T*SCH+72N+VTI)8QE);_)+%A*MV_*B^:%;Z*6F.;(!VM
MR!Q>U"MUQ5:?+XA>##,S[<]07M0Z,DHA9#ZYOAN60O4D+HCJF^@V;ICZM";Z
M%23'"_DRR*<OZ,C8=.DT:D0>\=(9U(1D[<>9GL3[OGI.%+E4@64_KP1F(TEA
M_GVC\+^X2?BTP/G39^:$K\ETQ?*Q11>W8B[.^^=D8D,?-M&L<T#!021.9W$0
MKQV(*4^=X?0)-RTH6SAGM[#JXA;,Q[FY6_PA0+;@#MUH"Q)-JBU(%!DS9`ME
M@SD>X_W$HWUP8K>F<%YI;;N5TMK4JY06^](^;J(A\2ZVPYF1'=OB'+)%("@!
M"_1M5<-$8\PZT:1IFE,5Q[Y35:3"^K>!>5A#Z6(G+5B1SL/H(<D"*1QF*B
M'`Z\/2^Z)M2J724UR^\BXZF/NIYJN]^L-P;:L)%YI8O]9+_4JK4-4Z.43V]&
MYJ16J'6.'JA`=*?"#,1VMM4?2_SE%.7S@@YJ5;1-L[JF0)O4^L<=W6'N&-6%
M3>`=Q$8B#@>(I`.U3[1--&LS6+0O#/U`:DFL[/:VK"9MX>2UM@"+'&!8/`_#
M1^9AF"(C_+YYSP@K]`*,[V`[+2@P"EP$V2`K`HV%VX6Q`V!LF0-CU?>&\??]
M\V'<T*=TT8@\1I-G\'8AC?;=%-)HWYV&E.U=`%*V]S8@G?;/AU37?\<A;0)(
MZ^=`6O'=(+U![_I;K]*[2$=R0G<E$$+3=9`V(F*1OA4(G_;?%,*G_7<:PO7]
M"T"XOI^;:?RV/#,#`&C!/$#/WTJ?O=*7[+-@WY`X=4?;[`\#YHK]23#AA76[
M8!8<N"F8!0?N-)B?^1<`\S,_)^J9JD*3OIBI@&]\E56_9%A]-?G.HTC)H$K/
MEFR;:(@5`M"/??<7JGPP"?2_#G`I44[Q7OSK]VZH6P[-;ZB.@RF;J)*@P`&2
MH(D6Z?^S,TXGR"J\AR#ZXNO*_*&;]:WC!^;WK<\.INQ9D``;)("[HV^4GPW.
M3X!S@+MQ!OH/+I2!_H/0ME-FX/6;94`>G)^!SP=NE($?YIE]<"#YS-H'4SZS
M2?@'%H1_@#0\B+X`HG]T7O07;Z7ZCP\DJ__"X(V*_X>)?.O!9.3<H1M'WC^X
M(.R#T!W$PE)VZW57PFUBJ.;;XTUK$\KY='#`AN0!=R.98]`+/F[J:BS79PI3
MS<CBST*6GC)D"G>2).CB)L@"%BY22*"OZ2Y3-(E&R(8%_YKDF%AA6T0QP0)<
MW821I`D/)HN)B174G#W^7'501@L.B!6+3@J)WZHNNZ8:<F#1?X^J'7(D]4&T
M:8UR6RW5/BMKNP'/L2NZ?2HY^785=S!'O*$@YXU7R@NT)]X*^.C%)_BB`9
MO55Z%;8YCMZ6/+*LNW0<[L-O77H;3NR*&Y$1N^,F;/2O.^OQ9U^(>\YZHE=\
MQD@B7W4#3OX<,G+3@L?_'_;O+PQ3V"5Y8(#=RI;AJURSOLI8M96IU)?KRXOT
ME4RYGC$;]#N5Z_KV"IB5[JRN(+-*!:;BBO)MG06,)F=1>AJV:.\/<><++J9%
M[Y.LL=68GYJ,WEU65C;""(<QMR4[,OI8*[V0#LWG/$6=\(V:`L.<M.:+PDZ
M'-S<A;A%&T?.X4E2NCQJAA*R:I9@7A>WP\0:=PQWHY:^+X)]#X(WY$KZ0QYL
M*;.[DA_,PU7-6F9W)S^AYD]%W!)W^A_RM>(6QG`8-Y>A)KD]TQ-HGQS/\6<P
MT<.X21<7H[E"<XD'YL\>GHY],M&6^)@+B=A)+,1K%D2S.;I4$(EI-FI2C*]9
M?0Q6'W*^5M2&6S*=B?R2%GMSB=,NLNQAYA^'0ZT7)#>DX#*]%TJ_'5OVCG5!
M[7=@?DOV"&?WH";<[LQ![="@.FC,P9#<F2TS+NWR6)Z7E]S@@R31-;4Y^G/<
MQ$TF5LZXM4MC&2"7W'`170NK0Q0WX=8NASD=6P0+=#2#2+/)]5ZQS(7J6AW-
M"`>VAC?V)H*]&PYS]D+R%MZ:1+7:4,L8"G<65S%.@G5Y:3537LM4[2JM+*U&
M-A_OM<+U=\MS$0*);<MS(Z/,.7HO<P[I3U+G=%]3YU!Q#[1\4`A""R#K%*Q3
MYZ8ZFU'A2094&&5<A/FD&O%#]&;-M\\Z+.U1-H@$:<$VWX^%%@J1@?QX4(@*
M05UWA9F5%Z$=N(CTO")L5>PBY`>`.>=4;-1SZ.(&9`"_\%"<(Z9%ZAZTL`/<
MD"!!D(=*.+N(S-X:VIF+:H/PW-5QS)HC0]1P1BR;##*&L_I:CT!CJD6<;(Y,
M3D["R"*;1YC7CV!>,@9JL35@!C&VP5`(F+N[Q[/\YQ0G6=?,@KJ+62.I#)6%
M65/).#F>ZY_FL*C1`!:V$"=9?!8O%[))_"CU'IQ^"L%@8B\UB:PDQ1SBH8BL
M)ZGW(+M3R':2FO@:U@1L(^]$&ZP)(5&S-,0CWF?#UI"`!)^5ZX9C69%9%E"-
M[$!ULA/5RRUHM]R&&G%9Z$F)978?A?/AGL`KN#RT4ZI(SL,]@5?QKM!34F5R
M`D9/X+4@M/7\HQRJ]35XFT82)JDFV;]H9QII8,GV95':%R^9DU..3,GY+)AW

M4I*L68)X`K>9".(U4HTF0Q>7L[A-E@>4/Y+%VP%OH88$+]4OX+UN8>]UBO=Z
MJ?XZ[Z0&.U!'C^""79I@E\8%=MF]\"Z[E5T:I<;K=D$\V08>",'%A3JPB\-1
M9/&9O$:[J%D#WW1>`4`P`/(J`!(&0%X#0-Y`C9(Q\2'@-4S);0:YQ2`[8SF(
M1S9D3VP\48C>3)Z2D]_478):.%&<'-A.[$@.[&+BP1-%\]7(3VC\";TZMH&R
M01W;B?[F$]M3Z9?,T2^=U9=$[3(HA'U0OZ<Q'QAEWSYV/C>1/YQKP#8#YMG)
M8T-41)/630N[D3TH-"+',#5$89LBJD.VH%"/!$W:M*]!D?.*W`R9$FJ0E<AK
M)2.V@W]'H#>QO,QN+ZFU6^U\28-=L-LBL<]#]E%J.K91JO)V2-5>MR0@%_*P
MU2]+/`BL7I=D(T*OAWW\Y:FTQ$6)3P%L!T%2LM)N908F2;%;$0NJV(.KE!J!
MYNVB<;7@1FYXMCRCU/N)/=?[C!#;\4Y_>EP>Z2&Z0<&#/*KG9.&X(6VGL<=)
M!<Y(UA1&L%>0U*/$SS'J`*-1[`*C,<@TI*:#3`-CV$U\!<Z`\']02P,$%```
M``@`,V/'&C<H/C1(#```\QL```L`$`!L:6(S9'8R;6,N;U58#`![C'XW(8@3
M+)P/9`#E6'U04U<6?Q#Y\C.Z19\?U:?5+KB*J=NUZ;=!P(`1+)"$+^'EY;WP
MPF>2Z>S6SG;+[.0&BD(3!4UKW672=U.2;K?)5J<RRHK3D0+MKMF=::?^56?:
M,-JQ;9SI'[7;KGON"UA8(U1K_^IO.,Q]]YYS[CWG=^YY/"@G!4@")`,4@#F`
M%$`J(`V0#L@`S`7,`\P'+``L!"P"*`&+`4L`OP#<`\@$+`4L`]"`Y8`5@)6`
M58![`:L!:P`,8"U@'>`^P'K`!L#]@%\"L@#9@(V`7P$V`38#<@!;`"K``X"M
M@%\#'@3\!K`-\!!`#7@8\`C@4<!C@,<!3P">!%!#L6]_SO+D(@7U<Y8^RM/<
MEPZB!*%!&)`L$!6(&F0[B!9D#T@%"`LB@MA`G@%I`^D`<8-X0?I`^D%"("=!
MAD!&0"(@'X%<!+D$$@/YVM,\WC+;"5@=U4YM;QM**%12LB(E-2U][KSY"Q8I
M%R^Y)W/ILN4K5JY:3:4K:29+I=ZNW5/!BK9GVCK<WK[^T,FAD<A'%R_%OIY<
MUQ6^I$L^JGOBJ.[:R[H+1W6_.Z9K[=`]=T1W\DC@CRPU)WV^\AYZ%;,^:Y/J
M0?5CV_.TNCWE%34L+S;9GG[F]VVNSNZ#AU_^DR\Y=>[")4M7K%YW_\:<K=L>
M>2*WH*BXU%!5RUD:6AR_??8/SHX#[A[O*WVN"E=E@$)Z9$!5IZGD\-ES^O`[
MJH51@WNNJRI\%L$3,HQ%/O5719)85S51&40\J@^?00(2L;[`T!#_08U!K,].
M0PWW^MM1\^"9=EH80"VHU5\U7*G,&*_!E:/70X,LL_BPZIN4FMAGKBI)3YP0
M=Y)!?740V;!P]8Q2H?OP!61GMAYFZE^0#.$SP?"@I)>7.^55<R>R@PTY@,AX
M#B,#YM71>F(<%9$=V]11-J08MJNC',S9+U.JE!1;+(F5&H>6AU*P0<<<Z*=U
MGQ^,9:I2`]=14R0+HF]"QL"<+JD]LO)BFSMUO.5B7^B3YR@G:NH?I!`R#JSI
M@5_-0DPOAO*(?U49]6A+Z!;#@A0=KPV4MJ,C9</8MJD4.\^@ZJ0V'5A11'
M[#U7-2N*;*)(/NN0(^%)%IC:6\4[/$N\M,"ZD\>GQ*M,L47F89,Z:L9LE'-/
M"?UR%S;0NB\/R8D7B=/)Q(LW#L2<3IS2Y&$[+<A;6">W,*NC)LQ%V:E;"!ZR
MQ7COK;8@T23:@D2A&"=;R!M,\>A[D7B\TC.\5ZF95DLY/Z268@?D6M(Y]K/#
M-;%3V`)G1A;,1UG$!R$H`0OT[9?"<&W$/%RGK)M2$GVW51*)B,[Q3",:"A5;
M:<&,K+A>X(*H'O(CTEB,Y:I6^)M42Z6&\"FD/_U1^U/-&XSY>D\SUC.OMNM>
M.R`UJ3@?-4BY\HW(&-?R-TW1`Q4(ZC0$=:[)_>_8AZ<IEQ,T4).L:YC4-'B:
MI::_[^P(L,>H=FP`WS"M)],!#YEI12W#S<,-*H6N<G\`KKW4&,OL<#8N)[=_
MX,;MAT46F,N;QMRV:<PE2,6V_=-NAFY+%Y!W$EMH029/8(.(AW0(-!;NB+Q6
M(*]Q"GEE/YJ\QA>GDW>Y2^Z4P?`9FER[.R#2UCTKD;;NNTODTJX9B%S:=0=$
MMKFG$_EU]]TEL@Z(K)Y"9,GM$7F+)N7KDIL4:3U6:*.$..BN]:1?B%BD9R$N
MVSTK<=GNNTO<Q]TS$/=Q-SM>^WTUGG(#C=NGT7CQAW32@>YX)P7[FMCIN]=(
M?QH*H_OC%,+[Z`XHO.*9E<(KGKM+H<\]`X4^-[R](AH@[O';?P4^="A.G/T@
MFY"U!"^S#W]T/WRS=WH__/Q@PAXHA[[L(`E]N%'Z_[B'Z!A9A7<(1)]W4]EN
MG:W[/'QP>O=Q'$K8>2`!/"2`O:LOA!,]TQ/PY2'VUAG(/#13!C(/0?--F(&_
MS):!AWJG9^#IGEMEX*>YAF\=BE_#*ST)KV&<_IX9Z>\A#0RBWP[1/S8M^LL_
MI/H?[HE7?TOOK8K_IXG\;P?CD8_WWCKRS-X9:>^%>R_Z*[X_QY@J)A]$#2>I
MB9]D+PJSS%>'7>SH]4B&RQ"@&I#)G8I,G47($&@CT:JC!@@7"Y<I)-`W=!?)
MFD3#SV/!O2(^)E:8#\HF6(#/**$_;L*!R7QB8@8U:Z<[8V)01`OU$!06K102
MOU===$/57X]%][();7]]7!^FYHR+X+H@;'$4V"TV'(Y\A>VCR:$OBK"M(8@=
M1;%Z>$V\K[Y&O>\N#PY2_PBB,#I1>!VV>0N]+3G"8?75M^#;],35M^'$MJ@>
MZ;$]:L!Z]^IS#G?:I:CCG"-TS:4/QN9-N`$G__3KV3'!X?Z/;M<1'X5MD@,&
MV"YO&;C.#JGF*U`M-D4V^%GH[6LD<V0EYD9'0IE%147]3,"+V6Q%<'#?D0=.
M+6[QM+`^Q0GV.-_+>WO9,>57HI:DFH5/8B%J4GV"K"*J]8V0FN-0`W!O5B[`
MG#IJ@0=SM-[7@1J[5=[N5G"*;'&WR(%-118;_&`./HG,119[_,??*F(&Z-6
M]S97$VYD#GAQ0Q&J"Y_:X"#G&!E*=RN8:UY<IXZ*H7E"0X&CP`$SY[QCD0O#
MS;$/6+^(K<1*G&)%M!M""P61F*>A.MG!%,L/P/)?K*L)->/&%&ML7D&CI:'`
M:A%U;B^S\25_TR7)#NGX=FXZO0]*MR5`[3O3G@7%VXJY[+1^%M4%AB">%FL:
M:H$>TTICUCH'D0]5T[A-M20RU\E)=G!$LFH;S0H]@.O8D5CFN%VU,**`><D.
M7W^K8+6/8H?MJB7P3$?FP`(=4I#9-/)-+5MF0-TL#RD"GDV!]5TYWB[>R^[2
M:C2:XK6(=W%.,WQ19C\?)*GGLY_O'V3.T_N8\RA_@#JO_HXZC_(ZH2>#@A?N
M*%FG8)TZ/]K6@#0##*@P\C@7<W$UXH?H39KOF'18V"EO$/32`C_=CXD6-$A+
MOL<U2`/JZFO,Y'PNVHES25/*Q6;9+DB^J:><4[:9.(<ZJD5:\`O%?)Z8YD[L
M00L[P0T)$B;FH@)VG5&;7\QHBBN9W26EY1I=5KX!GLNU^?!<5LZ4Y.5E;]FA
M*6:*]&7EA065\HI14UG&E!0PNV&^O(2I+\G;4JC3P5)A\2ZFL)PI*R_5%._,
MAPE-.5-<4KI;HYOP7L:`KRW%)>4)_8$CX@_<KF,V;V;R<_)S=NAW[RXLWEEF
M$9'164%;,U"E%ZYU%<M<>KF/\BDB:62@*5V/W(4^EXE8L/&X,C("(Q,86,_
M\^Q1S$EZ3R4V>XPPC7D8"AYC1\=0JON\["3UAIE7?3FU/Y&AO#!I*NE'AC+<
M8RP6E4HH&=[/2B:7R<GZ>8D;I-Z%)(\B&`SOHT:0F50"BSBH9/,`]2X4P2CB
M!ZCA[V!-P#QYM_*P)OA%Y4(_AS@7C\U^`0DN,]L!QS(C8UA`%>%Z5!6VHNIP
M(]H;;H:F4^3?)>F8':_`^7"GYU6\VU\LE<2?`YV>U_`>_U-2:?SYC4[/ZUYX
M/5PXRNXHU>S8!;F$%&O*RC<Q.TM+(+?E:U&EJ\99UQ\S2!7QMDE;DTC?C'=-
MMD]PU.<D8?V3)(SFW"7-62@HK%R".5*N13$0KI`JE0AT-I[+WFS;*OR23LQ7>
M<C4Q3JJ>P7O5S-ZK9._54O5-WLD5:D6MG8(-=JF#76IGV&7OS+OLE7>IE6IO
MV@5Q9!NXSX*-];=B&XM#R.0R./464;D"_I)Z%8C"0-1K0%0`B'H=B'H#U4KZ
MV`?`HX*-VO#C=JP-9*..,0C2VS]<0UZ,WY*-ORF^BK4R/&\^(`_OC,^L(BQ
MS<=SIZN1?ZIQQ_,GQCPH:R?&%J*?=7Q'(OV"*?J%D_J2J%H$!;(?ZOHLYCR#
MNF/'+F;$YODRX-6DQ9SNK\?ZJ*`RJ8,6]B*+5ZA%]3ZJC*\/%6%>*]0C01E
MTIBK1I[GY'DC9$JH0&8R7RGIL07\UWNZ8DN*+):"2HO9PA746`0+'XQ\X;<,
M4F.1-0C*`]F10Q)T:_XL<<Y6R>RT2;S3+@E.1^C)T:389Q*7@--60J)DINWR
MUCI^WRM#`Q[1BDWHLUWVP=)%I#;>[D:+B_4[`CNU=PP!'B/N+5`9-P,QV2
M.8$!%)R7%)S$33&`23"P83O\,>2`EFWAY5]:<LW!F2COH,=EH.D9Q#8KY3F#
MRV$)LN^`AW<@ [MT<#U+BIFRDN*R1YG*$CVS" IJ?AZSNY+)A3NK*<U;^S]0

```
M2P,$%`````@`UEW'&GLS>+4/`0```@(```P`$`!M<')O:C$S(#5E,#!56`P`
M`ZQ7-P1_$RR<#V0`M<_7;H)@`$!A6UL59`]!1$11%`=.W'N/N(TSHM&^_S/T
MMR/I16_]GN`<BVGYX^7'ZQ<K\`:\`S:;S6ZW.QP."()@V.ET(@B*8AB.$P1)
M4A1-,PS+NCB.Y]V"X!%%KR3Y9-D?4)1@2%7#$4V+QN()74^FTIELSC#RA6*I
M7*G6ZHUFJ]WI]OJ#X6@\F<[FB^5JO=GN]H?CZ6Q>KK=[4%$"?EGV29)7%#V"
MX.9YCG.Q+,/0-$61)$'@.(:A*(*`*A@&?:`2M(+B1_ACX#'RO?0[:/G?,_X_
M[K?KQ3R?CH?];KM9KY:+^6PZ&8^&@WZOVVFWFHUZK5HIEXJ%O&'DLIET*JGK
MB7@LJFF1L*J&GOW_"5!+`P04````"`#37<<:O6V06G,#``!3!0``"0`0`'!T
M86)G96XQ,U58#``#K%<W_7X3+)P/9`!U5$^(W%08_X86-_^F>A`IU$H<*R8T
ME+Q,,MG)>EDS4_/&C!LR<\AKP9/'4@JETINX*#VTL.JA:R^+AQ9I)H/T:3VU
M6T0L@@@B>)%>Q%)ZDM)2#R*LWWO)+EU*+WGO?;^7W^_[?M^7M)3]B@;CE/M]
M$GC1..'CM`Q[BT`5'>)!'2Y[E1?THCCG\:`4N\F`QWF]R_E$QN"X8L`XXUT/
MWRZ)Z_EP2FG#-.>!U^][T9IY*#[JNO"%L@\RR@,2$B_*<IY1R9/RK"8<\E32
M756>AYA[03"_8>'3AN^5%^$CDW+7O&QB(!IQ^2;\I;P"!;]NT8K._=(]$M@1
MP^.H&C5'V%(L*!*,%7-\P8Z*E!<SW%5%`@=4#YC`6(VQE#.)L026U#=AW40_
MI)!3I%%S"HCGL!2.JT<%GFSC2=2<))[`!RJ%$9;0F]&HV)W2Y^H[STRI5,>-
M;".9B%T"WZD97.J\UUF*5N&V>DR:06HSX([Z/IH05S<L*LJ_:=%Y;,,#]20R
M22+D&>+*0-=.`Q-)53<MBPI-&Z^*K."`=@;$-8S)JV]H9U'PB!0,M0^W,<<+
M?%C65L79(?"V=@XBR+7SV*ZM><^'L]J%G481+RR[GJR^ZU6WL$FUE@VKVF=(
M'G66X*)V$<Z;Q3>N^959\$(V]8KV)=)/<YGZ*LRT*ZAQ7;LJ1H*X\+-6/ED^
M\L=S"K]K7R/EJTCYC_8MH*5H+UHK;$9)P5M9Q:Q(;1M>T#>1'V>O\3B32@E*
MO:3_@%*N_B.L871E-(RG=.5=<WEJ'EKT<7X'^D\[U06$H-D4CNF_8`7T&@GZ
M6`-&A*UP2O]-1KL!D5&<BAF%3_0[<EQ9)2>N-F--_Q,S?PTSW]#OB>9G3>&7
M]0>8S:;^&&+"-RWAKQU-"+]5;^&N_N^NCZ+@-$(OKZ'OM9LRDRU];XOQHHH'
M"#)I-.-,&OVRH;88%?G@W.2X6FS&J"V^=QM\8U]KW<1O'J>!HDNQ<(GEL&+L
MEWR3I_E.&`>?S?>QT4&^R0[?I.'[U'!:ESJORT$KC?M[-LP@7"2>V9DNOY4.
M2;?C]''0;AL/]ZR3?M=Q(UQ\AS1_%1>[<M_X#[$0@SV!AH'CN>'V/^<P!N&1
M\=S>C9[7#T)HMW]=V)$8BR9O2]CM/Q9V272EA&S\L'UW04HT`J3K2ZR'6-;^
M>Z$A!_@?4$L#!!0````(`*](D2:>R^'R2`(````#```+`!``8FEG=&5T<RYB
M+F]56`P``ZQ7-SJC&#><#V0`?5`]:!1!&'T[L_^[E]RFB&<",H(H2@Y!FP@)
MYJ^PL%B10X@@I_'__S^:Q!A$(0@GVOD5B\=U&T@J&SG2NXTNI%'#%8(8L3@V
M38A8G#->;'V/QSQXS'O#(+\8I(ZH!&EL5PKI(AI[-QU602.>JSQGHG<^;R]V
MITZ^TIV*(T&,JBT0QC^K04T3/'PQ%<1!M:O&E)\.GHT_/5DUWLT$K\;G\YO]
MM7+\I!J(@Z'XFE^:"L2A4`R$XG"X-!T()Q3?6MGO9#@;3<:SX>18-I:<R$:2
MXUE?,I;M3H:RP61_-I`,9MN2?=G;1*0?586\UJE6CT[D"R_?%';UB]Y0>6D*
M_\P.96*K+_XE]BA;Q6M"F7":<(8P03A+.$<X3[A`N$BX1+A,N$*X2KA&N$ZX
M0;A)N$6X3;A#N$NX1[A/F"0\(#PD3!&F"3.$1X19PN/W:52N(>Z*,!1A.,)(
MA-$?Z;KZONTQ*B*M[$SCC@B?(GR.\"7"JDS+:$-K;:&TT.;646HVZI-*D_5&
M4ZE>:K:42LU270G-9A/U>KVUT"9*;2[/KJVNKLTNM]#FPA:@Z\7BRDKQ+U>*
M7$XS<,Z9QKG)P3AWN08^UV$>^*`>UBFQ+@'DOP.NN[&AI))3/V6@PDY9P76C
M1Y=EMB7+8)G@.DP'LLJQ(1.N>;[&+<W/:7(JYVI<UUQ/DXG-+(]QAYD6XQIS
M3<89\UPF$Y/[MGJ.[7#N<R<GBWG.YS+1=,>4:[KIZISI;D[GGIYS=)D8ANT;
MW#-\S^"^X5D&9X9E&SW_Q1]02P,$%`````@`,6/'&F`F&X!S!0``)`<```D`
M$`!S=')O:61S+F]56`P`=HQ^-QZ($RR<#V0`75-K;!15%#[SV,?L@TYGV'8K
M81F)T:QB;+0_-)$(HH:2JJM]L+LM.GVW*+[;\E!<E$F:C4O@AXGGQ\9QI<HL
MG=JJJ.NR$A(3.PC-&/^@I0E!RJ,FFR$A6#5QO;.[1.(]<^[]SCWGWN_<._<`
M+TV)&J3Z!*E?D(*B=)V?Z@Q(#T6DAR/2(Q&IME9:(4B-*Z>Z`A(7T:_RTET1
M?42T_B9K1D3I8LG8:'49<:O#>,;J-+9:,:/5VF0\9JTW-EBR<9_58]QCM1CW
M6D\9ZZV0(5E>XUFK;6I4-./6`9N(BT@UD0.=@18WK^MB<R*EBP0&[WA06A6I
M@N!-$+)!L5<@_5RY+U[E;5SN]26>K%ZR83),F;/),&V>3(99\U@R[#:GR#3`
M!JAJA5F)[H]EP`;Q_9VJ8R9Z*#[._RF3G1K_(O-"9=XS$\]JLVF(:TPZ$<^X
M"[_S[V\YQ2]/=0BG#DCUR>#P6B6JU0^#$DOV1Y7X1(S`3A4*$%7D0B(J_5:+
M:^L+$+.M&+$.10L0MXTX@LNVX[.GS)_>D^HG6C=/M&W6@DJKTD;X'X\,*6U#
M2NND5#^>@TY%SB4ZE>X<="D]N427TDL"^Y0!#91^95#[6<EK`>5X#ER-;NW'
MBB$%Q^>D>O.$;"2V#]S2A4N\WRB9S@G94*=I397+4#[XA_:\K(^*8=9TI!H$
M&8H-@MX@A!GK.B'I508)4;<B:S\H/5JO,J!/BDH?`64J.XZU7!-]'D6^L[LE
M.`[%#J&%B<R1WAFI`9T]%W99D]G@CC[A3*\PG2$/3@NK=><IRZ?19&0/LMJ\
M&I"V"O:E2V<%%<B])`_%"9S]/R*[\0RWO"[LXBB5(YY+PN:J9H>.#$F!2`MY
MS)>$#+7E()WYYVX98']L[J+`^\E*CD\M"EHH=5&87!1R\+T2S<'))Z+;R7=F
M43#IYBSQ'X[)Y9=!'J>=3])Z4G7/Q!LME9N)DPI0J9DX[V@N!8<IPSD=4IE;
MS1KRE/XS#QY7W7J?<"UNEY@O(FT36E:+)"&[%4=$'<XUE\)K4J.B/B9:C'14
MU'=6QUW5<7=E;-FS)$-BI[8GM9+1Q%2`B6JC"J77FIR4JC4U=RIHZK`07N;H
M%"QHB522)B7:(:1(R:X:Y[UZG4E.7F=J3Z_3/DQ-BM+J@`:J6X*(MD:MSY3(
MS=L_(),@@-2C%E'K,OMLW!6HE$=N6\"NCV5"TLP0BHR\99K.J*1JC8362W[=
M`Y'SJC6HP@<(,D(W0@]"+T(?0C_"`,(@PA#",,)VA!<07D38@?`2PLL(KR"\
MBO`:PNL(;R",((PBC"'L1-B%L!MA#\*;"&\A[$5X>]9,RV4:%>$CA`S"QPB'
M$280/D'X%.$(@H:013B*,(F@(TPA?(8PC3"#\#G"%PA?(AQ#^`KA:X1O$'((
MWR+D$8XC%!"^0SAATV1`$]*P(0T;T_!H&C:E(9&&?6EX)PWO7C$7[9MM(-<L
MF:G;36U%&LZFX9<T_)J&^2OFM>I?!JI4;>W9BE2']N)"?!S6L?Q"T=9\>[%D
M:WNQ/6\K%(M%R.?SI6Q%H+TBA;V7Y^<O[RV4H"+9:FNOMILVL&Q3T^G3364Y
MW<205&A@&(:F&,;)`,TP'H8"AH2%0A<NA,IR(42F[#"Z'.8JAT$US..Y<<-3
MEAL>,FN'L>4P;SG,9X<E5CCOG[./74/:-=(`^$L`Y:5$;<]S2\1A.VO(%@SK
MN(TEJ;E=)#5P.<F6X.1L1LX-Q,-07A_%N"B?GR)4?@_%L)3'2Q&/FW9Y:8:C
MG2Z:H6B/DV1,>STT\3@9G]L^G)NSD^+\]@G\/H9X*)9S$C;6Z2&)LQX_RWA9
M/\<2C\/A]CD8K\/G=3`^A]?E8&B'R^T@Z?X+4$L#!!0````(`+&"XR:Y(52Y
MP"(``,,B```'!``<W1R;VED<U58#`#MC'XW[8Q^-YP/9`!%=W=0TUVP]B^A
M]P"&&B`TZ2V`%)'0!:1W$.D0!%\44"E"")'>0;I"Z`*&)HK2J_2FTGM'0*0J
MH5_NG6_FF]ESGMUGG]ESSA]G9A=$3D6^!^0B).](`0#@+RHYN@$J,&W>)\^Y
M!O6/GA34-*N0Y_P#;MP\GO[F#J$<RI&8T7_#4N3#XN11H[MZ4!"&*!Z`;)R-
M@C?.'8H(!6"X!#B<D%E`>H/P4R`/R/P_ZAMA]#><'APNICNJ`N<'ZR9I_]\^
MW/M 9!\FB:H`!+]]X][$I3>KO!D!7H(GD<84$-W0)6* ;FKUCYY%C9[LXV 2
```

M?=?[?%M``?"_AXG#><&N9R5BX6+%Y^OBX>*_:L1NQ+TW5][_6287#KIY4_'9
M_R;#Q:WT`(AV16SE'S!XE.@/$5@X_,3=X5<,Z<DH$1P`)S'\6B(?/=RG*B#[
M]?\HX?[1ORP.`!$Q"8B2$TP'H;\%O4=*1DY!14U#R\#(S,(&XV5EAU<RE@'Q
M+@QPUY'OH0B:0')+L9@D6"F$$[5UA=X=Q-Q5I"81R9#&\ZM],><A)CXXWA98
M/M%R"4KR"5G+(XB[5/B+N)02Q-8^$$37R@C<:P7^G&MY_I(N&00IES3";9=T
M@H!++H'.)<F?9*W,7_A!_92A#>M;CK$6"PA9Z7U4@X-*^'_NV7K$7L$'VY+:
MSU3NY&_]D=K]<V^RY6UJ-U_/A&_DI(_\YU_+?Q^.P;N93>UJ-.&*K!3_2/*Q
M.(=LIK)_]/FV/;K?')X<7AQ&A%N%61<`._,>]%X8IF].FAYP=7M1V$,6SYV$
M4`EQS`=H>7F.BP%&S"QTN$W,F-AN&Z]F4F:MQL7PQX)APLP;:#+JU&%Z.!;[
M":.&_X;`N_TG/,)9GP(*!.4^8:ZF(F9;I<_B81[ASVU*]S],MAJ1K6R:*%$!
MG>W;C&E91,35(B;2]'G2C'F*G*`&4-,H8G/2.,+(N.+@1E0]\'!P)Y/ZW''$
MM=="]-](G;+L-+=UN8]^.C3WO>ZS+R]XJK]$=]$ZT)NFKV^H2-&_R\QFWH;W
M83S<%)2[?E[!V=]<Y]'HYLHP`>8RE'#)OPY))V7V@)RC4&J4.[@[+EH(0U2O
MO&-Z7$37^3WZ,WW8-,-O^%_3!`B4>VWHH960/";QE9!-'4@)M*[;"V5?&Q1*
MIF/QZ-#!>+".!1,)DF%T/C$^*^+J^%CJ:(['6N.=3)X5N":6">4Q+8'VJ<O`
MSDL,*@SOO3!FH;HK^LDN[&`S@#J'7H?04ZD^__M;$H;%L-W)(@PS2EPS-,B'
M3S3L+XF)P(+,B8=-E8X*0+I)X(+)`U/27LT<`1*^`.6=A[ZIC+R>VRYLM$'
M/J!HRK*H\W8M*X_"DSMWCX$4U_$>$IX!LJ=RGEP&]#;[^GGD-3:2XV$Z]\>E
M+H4!EE,(B<XURV-0'VD&YB_`-;3"->!YX-]KYR;K1;'SS?-K#G7_W^]\[=_O
MT*Q7;AHKCM)XN=5K7WC6O;H;O=-&C:U2YKRGS!FLS-G4PLCY=SQ+D@XJX5V`
M><'C79#G"14?S9,M4#2B@,<;-B1%?8\GKIT7(E"`XPVS,\FG\FX[OK#I,&[J
M<)]3P3QRK_LD3SK2D'K8GT68:4>`OMLN=9L`))3)9Z!`L9!#]9?([RK>>X/R
M9DM+GV_1V<5B6[P[?1I$YIYYKD9_2E(6<,]DP`.EI0IA%<1M9C4I/%%#ZMC7
M[;D8*T9&.#7K62<.;WW6@\-GG0W@\--G([@*UK.?N`KKLTE<1=;9+*YB^FP1
M5\EZMHJKM#[;Q%5FG>W@*J?/]G!5K&='N"KKLQ-<5=;9.:YJ^NP:'^]N4.VO
M-XK:I45_1^URHG^@=L71/U&[RN@QU*X!>ARU:X>>0.UZH2=1NVCT%&HW`3V-
MVLU%SZ!V*]&SJ-T6]!QJ=Q@]C]I=0"^@=O^@%U&[UZ8RUE[\BD`HKE3A+!Q7
M&G`.PE4KG)/@J@/=1]>?E+"Z9,%'X[E'RV@-FL,NIPWGBRYG#22@$#I5(#$D
MI'R<-80MJ/S$:[XQC;#GUSAGN3[E%Y)&X"K_XHHF)_J*;*]&7F_'M88!Y:Z7
M@9<C7[X,7[ZZ$J22V-K;+MM;?_@4H"C++W%S+(X\1H>.;A.#%`",&^;[&HD,
M)1$HX>R.VR(SY.]-[ONL12\THO=>1"])R>#,OW^4U5>)T\41O?P1O8\B>ET?
M(C"TI(CA'(.#G/-NATB9_AJ/BQR)%\8CEBS!S#5!E++,OF#0+U>`C+3@$\!R
MNW0I"4.I4#@GBA`(;5,G4G%/HZ%,S%.!Y%6S*6"!T(Y58CYR%D[E@B5!MK`E
M>'6>L$)>G"1/@20F.[M#W4.9)C%0`:,*5BU-(2:ES.>D)G+^)0J1X6-(>%P-
M8:'B5,E//F?%Y#L<'!-3TD*HY'&7B[734L:JTDY(XU136N?C0Z1UX?$5TH66
MZTO[>117O>IY*5<#]WDE5V6_3H+CV2%7;>BY$5>=ZIGPN,R@X.5`XH+[)<(\
MEY@:`V*@,3&*,S4JLPNO+FB/^-@?\:F@*Z*.[FT,CC`VN@4A]K`)\_`)$_LM
M3BWZ2(]8[$(<)OH?PYSL\1TRC3NW[L"#N]7O&Y.#V1+8<MDJ8\!LZ6PE;'71
M3FQ)B?T>_)D>6*1P:7K^!_*+R@)<GK/$7``$RA"$>,>3C][3`->IA"EG<IL[
M0I?3L!O"4(<'S-&S2RPE*12`LLOQ+\B!O"61ET1'A0%&Q`1_9Z5+>.[/L\?U
M['?I:^0*W_S!!#,_/-%$;YN'JG#D'H-MT53>!'K="IQ9^0$LN/S@A"/B_3Y
MJXP@8TDW/^:T^H`/(-L=D6?3=CT1&'&CR/8>32@>ZA-O7.`@8@D3NCXVF_ZK
M!67FI8;,J8#X(KYJ9_YA,=7VD:K!+K;-\6L"L;29(#8N.>'V?8_)3D`5M)+(
MG@(H]@H.ON6_U2:?T!+:I(QU8I&2K)66/C8%GJ>`EE>IG*D\5O<:.J8[9C_?
M:^DZ"J<TGOK@45`D@0MR"-!-MK=F+9R,<H4P9R3:.7Y229K^^.'+A`";%+5L
M++L]>Q`YH+%>4U+#A!'EZ>NNMW"T<`<H7US!B>@EJ:OU/IK_"QXWP9L(Q\E7
M.)GS?!8>L6?KC=G96;+;ZK+?\"Z?N$]/00]3DY'T(F8O,2MT2RV\/99?:<5K
M.L9[ZQ9OJAD+;\FM*%JS5R<SI5A!DEH#<V'>%*<H\8D$\\1[?WT?Q.%5!@NG
MV>W(]].H&/I++')"+6/R*).MOHE`XV36R*SZPA.4X=*K/YM1-S;;[/FSV=.V
MOX-)/I0E,R^3%0+JC[)Z;PIB!ZH>Y9Q1TK_NY1-3PX?N!^/=7I/3OL1>8G^1
MN)4F;_T0I-`6NE_:^5OW?N'O!YD\/;F[=KA/@;`I7_SX\$<':TO%(=N#7ZKB
MY%1))+X@0<W`.(+-J/"W7B,?_!2H@N>K'.AW+QI$[.#5_;"LF@*4V)@J5`[?
M,,GQ>64G(!GIDPEK'KPWM=YB8`%%IGT8A.[1Q67GT18`"</^KC%)2IIKG:JC
M]U5"5D5_!9J*/G]&3J?^KO4U7;AY'D0R6&N_;-R%_JBKA-5H0",S=*>WY'X(
M=NXIJSU]4_Y?2;$(VQ^/2QXF@2=5NJ1-O9ZK?28$UKW8GEW>GN_!/\;B30WD
M6<HG4,6(LG8@2-/+<:?-X(NBI_3'1N$X(G+JM!<<'`U<T%!H6"%-S3I\&JJ#
M112E"6O8LU_EWJ=Q0*?LAA;\%FYC(YL9>N9O1-ZRJ8KD>_2/T\UC[\9N>V6_
ME3;-%4:IXD.I4339I5LDPL]NE\U[6'*_\#`NZ@'(,(&,'RL%/DAW)P?8<*8"
MXL1C,;/%-BR/P1NW+[,^2%.K[4Q'=AU*=#Y9:*,EW_:M<$Q[BJ6MQD:G*[4^
M=$O-]SK-FFXZ;:K,,H&P;X`#$`(JP.?IS'HQ:!6```DF2A]@,51M64];PS='
MZM@7KB"H>[6[":R6QDSO!M-\$>.#0Y;22F3J$ZC#*DQ809T*XS'%NNZ'CZ.X
M);$D[DG.']:``U2I^M;PF^:^IIMF1KLX:(,712O*QS([.JN_4?M"21%@OB!*
M=#V2"+#S%_\VTQ@^O)-+,D*Y&?V?N/_$H=CZP\$OT'JQH:>*"^$-8F/63W47
M\M.:G3U?D5E,G2U"(,EBNWS\MV\+"`H)"8N(BHF+2TA*(:1E9&7OR,DK*-Y5
MNJ>,-`-3,K#Q"",4U71-;#Q\`K$Q*=D%'QLZ^G_,;!^>$S&R\MDE0[Z57LQR
MQ"1_+62-GLIBBI9/C/H:&24?$OG5+U+^:7[Y?T;M:N.4J0(&*R^'!WAF'4]/
M[+,!A%2-I$25>$6%*%[D@[!0F6"IP/O;Q<5\1;R%/`7<^?`\+AQG+D<.[!W[
M6[;L;-8LEDSF#&:F=&C:K=14QC>,#(LXP/#ZNFM7,6=HA9#?]K.%8I%Z+\;/
MM$5AD6\OW6_H;K=?ZI["HM)>@5_1GO5B^/M@Z9:RQ>=[E7Z+=W_G$%[Z^;;2
M#5^HWCV@WNLR;9E>C!9`\^<0LOU6`A9C!-!".81ROY7ZQ5@!M%@.H<%OY70Q
M3@`ME4/H]UM56(P70,OF$&;\5@,6$P30"CF$;;_5^L5$8?(<PIG?ZCXU'=W`
M]9=E@C\+R?7LH2W`(/8)!$.3>Q.D>I;)N,)A:!9O@G?/"AE7-`PMZ$VHZ5DE
MXXJ'H>6]"0<]:V1<R3"TMO>I5,\Z&5<:#&WE?>K=LY$G>-%;>*26.!2#E!>\
M^%5XU)XG6QX=6\N4'<1L&+$0(3@0Z1[.<B2T8.SS!YU6VC!R!Z;JV@C/ST
MH(SE)2`4A`6_)@H'^`1%)67DE50TM/6(-,.[2;4B>LCN1[9_`3OI]3M@2GX2
M!W&U2NN\Y3[>4\Q<A&2>BV=Y2Y@2MXD"*IAVD&IH!U@-VTFD_KJ+6"/L&\G_
M+]!+[EZZ)?2X;!L.=\A;YG;,7^%Q*ECE=2Y<XW,I6N=W+=ZX[5:R*8!Z_TL[
MJH]")[J?W#7*-3=8G]_*+MB2J@T0X8"OE%GT4+?=35/[`::,L*$3-%%ZG2&B
MGI+EJ6(E"AD(]EKH^*A6.[78\;'CDTEYH8AL`+KC\S@J65M)Q&A71DGDO]T+
M^4ZQ09'G(FCG'587%0&-@=0/"LA_S'BP`WVZY!F)T6:DD;&)<-I+\WB>Z@0E
M]?6.M(SN"QO@<?9$=+@W')$.1T4/QRV4#UU,[B '[*\XQPR+2RN&N,.3T[L5

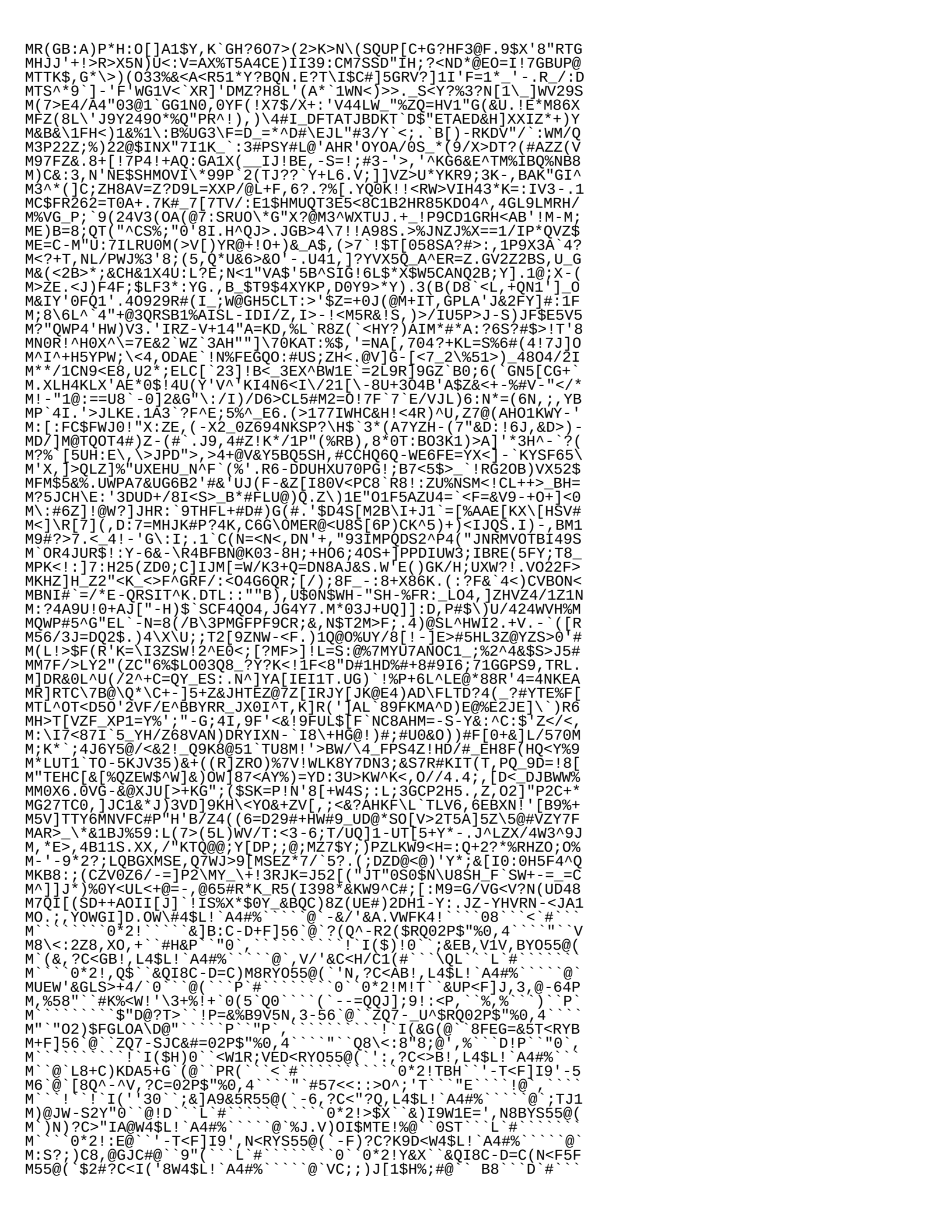
M+=3C&6K3(OM'86.5H(UV!T$E%,NI4V%-3Q'0\43>7TVTGNTO1MJ!2A>?J<IJ
M))E5,BT!,@4\<`3SEYXJA9C;O>:!2=HSVL&"-/LJVD2^0HU8;&>\?1XDZ[OQ
MJW<!?3Y(^AX.Q*+6!6SGK*T>&T6D$<O$I9%K%/9VBTG^'EA>Y/O#'E)H(+2O
ML(<6BF;JTF+G(?'9'XAXZ.#N[J#D[%7ZFLA-;23?2,+N683L'W)J1+6N"E7V
M&@022@/4,U1'ON(0`5[TPC#-4OG_@)Q#-[Z[JN9N:W=5VQ9XJ#H'1#1(XE88
MNADK;O_$.XYVZG;!(F5V1*--FWZ\SU$T3C+(-1Y-.LW.>88+$"DMG*G\G=IM
M\0M+]V.&.81VQKU#\MLG/9]8AVZ:5PT63HS<UEAJAM<VI=@ZHZE;WXZ?5B+^
MO>9$$+"7_?\&7I]F?#EH]]GB*F)>MRQ^S.+J'/ZX&.7J5#Z<8$`:&F,1[_-
M0F-ZAX7*1-9JOJ-\P=.)0_-3VB.HA69=FADV@0.1BW@C/U7S\U9^LOF!^H"G
MC+)^@IR_2--2E4I%"IL=6]MG&G*M_9MY3(3=4X2T^J$];Y=9M4$4F#/-2HAR
MRURDI$2C<H-@\=MMSBW?WTEQ\)&A)*/V9!7(*DJK)G>)<G,9(RX_I4"?K)S,
M,KM%WWKK[YW8,^<LR1:Y9-J5M+?"%QIW+"CQ:VL'KBS%KE=%ZD_7U#U/-621
M^_0TE<@=R-^?[*LVT[5HJ3F-:UWVK_=#P.P;>X0I+S'VL_P6[QNJ33O8L:6%
MM[6[\\65^HO65#G#JR=R%],/"^(/TEHVH.^SZW_?WRC-.L\AC9MGN?P.*U$G
MT'/)A71W!=/U<)8FP)[`!GK8UOW@VCZ8YY%*9)WCGP>]<Q[28]B9(ULM-O>6
M]T2M,P,'>(+E7TWS!)$'LK]?==R]%;H=F=^MG^\.3^&-/3E30NJO773YL=ZF
MZV-H91H\QE\1V4$/JQ[YO^Q3UD?+@V/[9*CB2AKN?^4%G-LK;WL=8/B=T)0I
MKN='Q+[\?P<94W%K@W("YP("YV2$H9.9_F#45>/:%%+O"OKFTO=<OKX4V8?\
M@`RZDDL[&]G4,1Q!MO8O;PP;#G.^F!8ZE3S:`/K1O41"<[A]XM`V^LU1DDK<
M\BAOEJ$,18/ET2CR,,]32`+^GZ_N59(DONIW***I35T3T=-U4%I(1T7F&\.1
M(9J3X3B->?U#;720%7D&[HNP*`,\PAV32$=ZBTIMCS"_'L#'O]*W/WJ.45M*
M;"_E6JZ8^"M]VT>;)B#P15K6N!UK_?CPI.AG3FN\K5O$B^R>[@$],XSA^P8+
M@=I]R_!=!8WO!QZ1%VR?!D_[X!);Q;EH&]A+K'M67HR9)Q.*#\6$HGS(0:=K
M"4L&6?GQFWR5)8E-L7RM(%>;R9N8=)%D5D[W9*80&='+XS;,M)#<.[+<:OZ`
M\/0'XSL%<_5129!PFH6*^E2E"9CLU[H/W5NZ3%>;*A!)MFVR7XY>>7MJ/PSY
M.-KG6`V@SJOGG4/H\7J$3$F2++&043;K=E:`;=QHW9/0R8DV1,^_NN)C06;X
MZU)X66:Q@Q#10$:6+WL-TQ;3!B7\K@65EHZ%3K1M.>1[H9JJRP=B$57%#,FH
MSCCB;/"/6'B=5`RT-(D^3@U,0R5YY,8[E)B6&\,#K9378J0?6C/O^,IJUE%/
MS&,J'<J57VDPCW?\L^)ZKC?>"]4/JD"`_T4\&<&2#PYZZ(;;US3R^VRMZK"`
M&``7+J&)_/T"]VK(0Q$3Z:00^OE8J;3\VKC9Y?)V/19]?E<96);F8UDK2H6T
M`CS$_U=U:[W49_?N[P[1%$YU:\]$#)K`W[.G/^849^4CU!PPY6Y9M8]N)2CE
M*EY0DK,$K&MZE0$!K9&"Z^I/N]22`N-$[<O&2$AIAJ1TY_6&Y>*TWFKSY7O.
M/XK,:FM[H")L%&<<HMGL:AJAHX&G=J)YS=MFF=75=E>%-B%J];T#=[]\L0K0
M'L_A9.9/[0XV'I:2`)0;0L^7\/O=%.N]0T:*60S]*=H_MJ(F="*(/X7!K@VX
MPUTBG,*<-P-7,LXJ>RWT"*7-\!^!`N_7A).]5*2'D::'4*MDVY\H]B-8QZ8M
M`'N&VOCHP)9OY*4M/P_?/U8;I!/+K52<^M>N_J]*)V8?-Q`WY7)6R>);3PKE
M;2&C/E5=U5BUY^.<)S<J/2Y]-&!G]*QF@C8HF(K#[942JD#+]L"X'X6_JC%J
M%I#^SA6;&T'8HPS7<N"QU-8T@*L:6,/U#4W,5/4$-2U6VTT5"?NF9G!##0TA
M"?5A!N)BVT,=+>O]#4M5:]-AER<+J_G&2J&X.P,_3!GEUW4,'L!U!ILF3(9=
MGMO='?4=NBOLZRC](Q;A_TEB>-RPMWIE\0.ML'84U0-M^@>UIS3FLG+"R6/3
MV=SFRTRD;&J.%FX*V+02,,US?2/`&+2CPE7U3!'NOEK6XL+M!L5^L_M@S7K=
M4TO[R@QR3W\IAP<?,R0"BVGED,&_"_RXB=?02@(52<^)M^=\M*9L4"WG8N?H
MS)F%*AY*@ET0J:JZHW(\MA.Z;I1:+&XL6I1NNEHBWP'G%/VUCN2]."Y6"-ZU
M91"THONDQP<6UNK8V\M)6"SR080'^)L@Z'L'])BPPR5.0U%N>LDEKL-6;G?(
M$]BL]A$]2NFBUOMJWTJ@7M\JH2^_?<F*U2UY4*0'5W^WC$T7UDI(NY.EF(P<
MZ'BO)+1E)I3Z^%93DXAO1Z7SV^6)P5DP6PHEFSW_>[YMRNQ\1[!B2@@_3SZU
M\"!BC>=2JPE:QJ2,BTT^PI=PL?4[%5D>'CCL]!1%P/@?`T4U$)IA@W%^=./?
M_IJ,`B!@TXFZ3:_#?/E$NY-\<2A>W+-3(OW:)-KS7VEZDV#ZX<F^8&/L'.YI
MC.NSE<<AT<5[8]F/+LN*%RF*?S=?4."7ME&9DGWIE&YGMN?%U4/:KNZ:%FZA
MH$J?NL(UR^(UR_=KEF5KEA_6+"O"!UP?^7M:)JU$@;]]YOE6A;1YD4>?/G\E
M*6!'G!XE^?GJ@=6Z?7Q08WI4KMWO]*@KNU.=*O"LAL0'5(MNTX3F\;'+JC[O
MS^]O*\"Q&LCN4:U3&8;@3T)LU$2G-PVQK@PV[%"N7"\G9XEBGZJ`PFW:]*^M
M3MK/3P!4#9!B+6],K:7^,(HG>.=,]RW/?$Z?GOE".9`;2!EA*_G+2?*7)=^$
MJU6XLTB^BV"9=50>*CE^'Y)B*HU`#$BS2S,@5B75$.;N?J;2=C%$(BIHJZ8F
MR4K)2K.XLM='QW%&\<=QSQ)NQ24E*"R2KSN$E"9QT99N-QG5M%3_D26%/5$J
M'?ZXY5V]Y?IIRR=ZKI5TMJ,CP[+G8,&<7J^>>#7:R_K]5L#&@??DH#3*F6L%
M==@S^+)G%=F'Z[%GO3B,],UXFR'S]C6:*RC=CND_#D,?/Z&#"MY7NDL_0+PA
MNF4]$GR,&9R!G6N8!WH;+*8_HVX='>C]#G6]9Q8JD5VJJ0'7MX:K;3!"S)+)
MT<@D4D-5#4V3Q>OKT"DP0$K.P');4LW(S3^EM'N9R)CRKE[."#PE"9:F4\>3
M084A#B-6=XG&_SR%"^O8OO>Y1]Z?3^38)6P6.2N.9<#K;*1R,U$!Q*40_V)R
M_90I;EY-W+9&%6L8;33]/9_[L$-MSN<#(HQV9L(`$S6/#?%+>H,UK,"0+SSB
MDQ-$6=0F7*=9*$V5ND^C$.RH2P(0@8T$53)_1.M!QE+[#P3,\'80L(RK3.^C
MHV@AD@[<$YW'DIF`IUIX'^3A([D4ZDCN&;Z&9'<52$^".>]A;P[Q7<3C?*@&
M5@74_EI<4,LEI9V&[<FBB:8\'DZ1=W_GC>)&BJIEO'!OA`8/P`E3M$KY5_8N
M*2S`R\7&6%M%7DJ8CY/U%H2:@I08#)H#0L%$Q$>_XZD3&:`L1[ON@A4(.:4.
M;0.S60_&!Y%M@$I8'[UU\:'NK0R%.9P:/R5@Y1U3U#)]S!3J]$"&GRG/'R"'
M6!5%Z[^9@KOG9VSK5=$_@<MHSWLFR:^GJ!T]XNE-,%,<!20=4T8%JF,L^:24
M-6S[9`O@862QG/D,$GK;Y*H(`1:5*".^_M&\FK$6M&#$&S1\'XT&QX-X9'1
MD.E[=!C-_54\XS\G"LDI0-!HJT\[95X@!18I-1=I<H\2R#S0+;JR_L2`?8O0
MR9=/Q8ZDX+":%(-%V>TDBD]P)B2"/8[@/#4)V^A)'KZDB\<3&A^AU$4`3-ZL
MAJ;81$$(N)^[H5\,"Z,(8AEG,*3N)]<B[@2K`FTAUVW1='JZL_?W9VUM86
M%KYY_JUKZ^SL^VUC(STS5>1&CDB`E]=)3<B!Z@HCRY*2]\C^RDI^L[/M/2N
M^?CVMEBD+\(B6XML1RZ'$C:O-FM]%^MWYD^\`K_<XYF^;IST+=DK]!MI'/%+
MW^-8O+.7X7_H13B)?ZHLMMK91)#+/9^0:VP>.EANJ5]<"+8(N;5P<A,[]_GU
MW6]G[_TMLDF`<2T28#0++2%$X0<7M`*XMM6`"]K[B6WK-^"UL(@,_7Y`X##D
M"IV\@2JNT/F])7\E--F;UM#-`P)GRQ598VOHX=Y2<".!J_**[!0$(B&AH*"A
MH:>'0EE9.3BXN?GYA83$I*1D9144[MU3U=34T='7-S8VM[9^],C1U?7QX_^\
MO5^\"`@.#@T-CXZ.3TY.2\O*R<DO+BXKJZBI^5Q?W]S>_JVO;^C[]_'IZ?GE
M]?6MW=V#OW]/+Z_`I)24M`P,3&R<G#RWA87%$7?N*"JKJ6GI&AB86-C8V#FY
MN7EX^;Q\&8C&8B-B$E)2TK-S"TI*RBL_UM4UM'1T]_</_YB865A8V=C^<_CO
MW]FU$8N\E7]FYM>I4U8%ZX"`K/KI,S9%F\#L[(:9<`:[#U^];9R]@"DIV0:]

M:YJ[Y+CW*#BG>?Z*4]D.G=NR<,V%M`_!M2X>B5R^7D=?_M)'7RZ]0%].Y:`O
M1[^A+WM_[5^&&#S7?F`_QFGYEZSNVB@X(,0(G?7T+UE_78A18+U.C<ZG[CZ=
MAM&?.FVS.EVK.KT[UH-_=48O=28%%8=G#+8-S@PI#=D,A0T51W0-;48\1@)'
M8J4W:K?'SC\CQ0QG?<]VD'O/B*Z0UX77_Y#7O%=^>ZVEUZ>)UU'7.>>+,\A+
MY^L?R$WTT6'B-<OU?N+I8N-YX74=\OQ&^*>E]'HK\>I_A;ALY*GS=<J-<'4E
M\8KE>NE&^.6D\/HE\H3WVNN&GIE.O(S9F,M,7_C/B_53PT;_ZEOZV-PGJ$#K
MH;2>.^F^_WG*,?F^I/>]-'R^6YLC,IZK0SA""E`0UBY>HO4,.39Q+:BU\VTT
MLF5S]?P0;=B*6CT_0R-;VXA(=?024\:G8?8N:6+W[0.2/WR;_T<KJ&+S'.\J
M^Z\GRIIUNS%8S=WK^!P4F7"++>^]F'2##OB.6W;_M81S1@]!T#:N[:^J*#^
MGC=#JNRDU5D0!XZGFX%+1%[#^)&'W^M-]L9'JJC=XT`@(IZ1->^VZ,=ZI(ZX
M74KWB<C#Q-9]/K.HKW]6&>_[E0EVD5'JFQ&_JQO=(F*7U4>%9-4.K%^(UOCH
M%>I:.#\-BDHKJ&P<4WY/]L0X.C$=QE-:)=O8QZ#VLF1:\$3QGHIZ1/30+2#[
MX^#:-;/4`^?@M*J>I5M.$SP2=[7-'+T'(]_DX>O'Q-.O[=O@45O&#6S8U0?X
MI,*2;3%)WY?U36TR:3_BW,VTE&3$A6_SP#EA[`/)LVD[WB&)WK./"&Y7S#17
MA7=/H^@,NTB,0/[@X#`J<\?Z91;U(A)I2N<V+L!96^3Y)R*;S]6_M8F6V)RZ
M/R2;=X)U,O<MO]Q.(&[<7%V"68V;T`[TAG>TR7A.ME>\C7SA;*HJQ4T/[*^,
M=7\MSTT*$^AT!A<8'^2K[.<9T;4^%5_(-:`9BE>Z:(K3HQIX^U!XNR%0EW6K
M+$"'Y=?7*'=Y$KOF)L7ZKW*RGVH14A*5%?@/PD*"`K?Y^7AYN.%<G!RPM]E9
MF9G,3-#4-V\8Z),2$VEIXN*H8F(HR*/(2,/#B<.(P)/13Z]/-A<G1WI:J][C
M,A(C0_R>HNPM#>\CY20$N)CI2*_^[4J#/M?65%5*"M[+QG4>RFDXQJ<5ES?
M-[U^<*5M5'T@E?R#P23EXY&T:=S/<UE@U4B"1E5QC*JYN=9P^.51?,+0/9=U
M.2_\S[J7:'1$1$)"2LC6E_\:&F9FMJYKJ@<C_OM/&L[^-F[R@^S"T/-]:<\@
MT8E&V8(A5_'+_B&[6HN=UN?2YB*M+Z1-8UM?BE_.-)[X/K]J/,'N+8BT!LCI
MV<6'5%[N_)Z\;K3?:0U2\E4['";JN;NX<[)^LB9VX/*$==_GG#P(U@H[O/+M
M`O5`!&Q5PW.YJAI1Y')<50W<3\G)E+@68%SO]D-0Y*3/0BT8V(01:KHV'C[8
ME(*/#?TSV^>,O-+JIIZOWA2V.L;Q*9D]"2MJ&OM+(Z;I&E[\;2-/"B']_$7=
MR:GBJV;ED-:JQ'RKZ^9`9I)&S)\ED3G0YW.JV2[@RSE[*=#@C:[Y=O7.`0(Q
M^DZ?_,/D#:/IF-FX>3I3!G,F2Y;UC,WL.]B\'2?.)92,BIZ9@U]$2@ZIJ6_Z
MT-G3]]7KV+2<]U7UG4.3RSO'0+A@9FSO6((:S0KM`C>-FN(F0F5_:5MQF,[9
M0(KG-M_0$U*2+PQP!Q6NQ=80_:Y"*>&Y!_)=C=ZB.QAU^]TBL97CU=@!_V;I
MV,7W5:T]DXM[1WLJ$D>+=E^N]_\'4$L#!!0```(`-5=QQIIZ_YL?0```*4`
M```&`!``;&]A9&5R55@,`-6,?C<"?Q,LG`]D`&/D4.3@8G#<Y+C*T*K;<9.A
MPG*%R4HACDX^KH;&2CH6.H8,CAPB#-T*CIN,P%*^`4'^7H;&"J:N!@80^1@.
M.;"\,5C>W\G+V$7/'R)5R:$!EC(!2P6'!/E[N@3#)&=P&($E3<&2/IY.QBYA
M1K[.,.E%'#8,\TP,+,T8&!@`4$L#!!0```(`&]*D28*MS<TN0D``(`9```+
M`!``8FEG=&5T<RYB+G-56`P`FXE^-X*F&#><#V0`E5A+<]O($<XQE_D#>T*5
M5+5^4%X\"5**MP*1D(4U2&`!T+1\3.*M').*4Y7DU^"OX$69#D72)*60/NFG
MI'L>P)"27;N42**[O^[I[NGIF>'O?O^,/%,6YS\9?25QXT3I!7V7(&\]^Z28
M/W2[0)!TL;I.CS5550D!;+%-7Z8J?=K!DTZ?OL"3B4^EDSCPW,;G:0E/'7RZ
MC=T(GKOP7+C#91`#H3$;VY4[0$IC%CG%K7+*0.K&2?"9CK+VWKE`&/K]F:$K
MYU?@O)($B>,3<G_V:30(E<0Y]UU"ID,O,="-8PO\OS]#VG-\4%=\[]SHDZS?
M!WL`8,CGQOT9L)3@_">WEX!-\>1[<4)*UV=@IH5@8+F)*U`743!@T+6;%*-H
M<0Y@IH5@8"J]412YPX1KD.L&)U00>?TH<NZ^32CT^@`Z!('PP1LR#X#/G*T5
M!3H,O&'B1DUT!-UE^65:B"P<OS?R'0C.&?:5V*T'&#A)Y+TE2R_I73*W014U
M*$<Y42*8B28G3A2,P,#;$^>M%Y.=,P8=BD05(!L%CKQBR%7@^P`%!`*1>H!\
MQY`WP1MW[?6Q(A"&<&2)X/#Y-@2AP#TW*&\21&,GZL=<<AL"'\)'Q)W;XUP!
M0J.+R(/Y<!(O&"IOP'(0Q:AP&S)XHXO@)[X[?)5<OFR;3Q$TB8)A4N.H"@R'
M>5\ZT2!F;`$2M0+3WW<C;_A*"9W(&4"AP8@@N/-BH24,")4W7NR=>[Z77,DZ
M.)=WGCMF"D*]T7''M":4ER)GZR!"&."%*BV)8!".1$$`@HWFBTS'I(R<<>;[
M3)/;H*L$^(H#,_B8`A8GTVC4:Z6ZLF4]5N"K($$(U^-66*TDHFS[:(/'A*B)
MTV-E0E6?&W20-(84@_$+$"I/PL"_>A4,GQ)H@[1''*'$RNK@@I,AFQ4VV2=__
M_9_I<:<#[3`KBM5BLZTY!N?L:DZ;<[[4G"ZYGBUR8-6</Q.@EJO%IYKS"UDN
M9KO)=#83+&B(RQGH5)]O*6F1<I7];S:=5S6K0^ZJU7JZF-><G"P_,P.JCLWO
M;SC'Z<\C=P2=D3Q[^>T7(8D["&GK[-R?)>.`]5E"*X6R'9E-MAHRVS#.S!NZ
ML-Y[01#UR8ZQ=;+5Z8-)=NRAC3Y`CC=[&\_<@SD('2]2@@O8DZ!$8`=:Y&M4
M43L8Q8*50.3VP#PLP&HSG:-T,JE,0M!:@<;269FEQZ4*&PML8%EZ=*R#<NXE
M`R=4!A2QWM0(!!_!`#7BY$<%8H$YKE$=8;)$;C8O06'2$<95V(IR9_A:T80&
MA3&['6%&U3L9;#"P$%Z[5S'$$+H.=-_TTWJ5\MF#/>I"&8S\Q.L%?A"]>/&"
MI/EREIY>>A&DN0ZK+4;6)!_;1,#08N%'1>#?."'ACOP!$OD@@2#:I4<_@HAI
ML8VP5ME4FS+;U%I]V-^%"A,AM4V/5`T\+YD)MCVB6]OT;I'?GV$#J?<:;AA\
M7G%U-":1:$U_:&F%EE;RWL("7RZF\\9!OKFA:,=$SS4^AL5W8)<"3GV->K(H
MJGEKQTP]H0I/D2RK'4^\KQ%R&D/K7E<%6`A\KZ_P)A&C(`F9S[3V>?DXB2@?
M%NPRVT#FETX"(P_YX80'CB(6K.CEX%H0A/4<,I.$98:FL]F:L75S(._3W!3K
MO>ANE-3=EHJ:)LM;K)"R`6B6Z>83\*U.Y)J[2`\>D)&QXR641U<@Y*KZG%(F
M28N;97KTO?,]9X:XV_>'@O]NCS*!?]GP8]@$_<%=RQS(\&-!??*&=?0OL2L
MD7\4S$'PYD(P?Y"8YX+Y4C#'L(0$\YE@0@$(GH(\Z+PL6FE>UF.8^GHW/<@8
MRY4XB]6+`_A@E,TXH4.GQ:R``8HBI<4%7(R'<<M)GA[K)F7Y6)#,+!R2]NWP
ME!\H\80W>O0<MJ])DW^@1U/?:.$A:U\)<RMZJ"T4"\H_Y_Q)ERG+AZH](TVE
MLX6!BV>SR]AB8!T.5Y18JL=M329TF5!DPI0)2R;:,F'+1$<FNC*1R40N$X5,
ME#)1R<1$(FPY'EN.QY;CL>5X;#D>6X['EN.QY7AL.1Y;CL>6X['E>&PY'EN.
MQY;CL7D\T_F.+PK6NU;0U$B]`4%W'OI7<)AS^WCKZ.%1,!CV7-'MZYT2SCG2
M?)N8GRT?R=1D0I<)@Q'3^5;V86];Y;T\2O@6>H1'B9MZF]T_$>C-EJJ+PP%<
M/$]]G;,[TNAE5R8RF<@?^`7A@0LL.P16/UTJZ(J\0ZXDOC[LUQLG[E0I^(+\
M+3]>"7HGT<>JAJPOG$4>VWS6BQF]GD.D'ZMM.CEXD7*Z^5BM\.RF,81E99G\
MY@B=_@1`$?MRRR)?IM=X;C.8N*K*,L]M6WR3+QD5FTQLVWE>EE4EOE$;$`"P
M2/K7]_]*\]RRJFHR$=\$SM7K]?J&_MK`7:PJRV)R_";Y:EK\%^0VDZMJ15^J
MFM,767^^S5;5FOY*P=.09?*;3.=WZ_TEPMNP;+D-_FXF%%]C><Q+R$^N\S+
M'`*I\I)\7&]6TR7#:&(0597?Y$["U+G<?_&S\@<*_,[D':JQ-X@A'L+'),C
MY]+MP^T14.)41BOH_FP>1`/'KS=QY)K`'0W.77K&IEM5_(@`;T,Q7H*6]&I(
MKKUQH?,N@5@O^U1/^03Z__!O(:!<2?"?/0&7HF6\C7'#DZRH-(HS9"?>N!&L
M4)?[1XOV FSB0?L(63GS.%M:2V A,A)X:E!G!AD&JY$]&8#56@;#&#*&EY2`

MJ0#D,%.&T5D7(`U@AG25X=.CGRH.7&L&(?VE@C%;@(E#S.NE%YZ\]E``MYQ+
MO.$_<>,8#J8>M,8K@,'2]95Q$/EX_72B^"D,(([(?(/MJ.Q&+@W`)-I7)[@M
M"P+H`\F#&:8X2X5IP@_^IUG-OWKRX$-,`@#;P-#;3(G^@YQ_'-"-$OYU3534
M]P<X'%"C5;-`OZFWIW")N7`C=XAW1A9F7?<3+X+):.^%K[$OG<B3:?*0&],:
MMZ)#NO0VO]^*!6(V"X,]B*(U]X;BI5I7Z5]^^1/$VFG96*7_@!E-]5;Z`2[*
MP$W14"I&?5"N*BA!1@_$62-NM[J'8MZ^!:(+(ZM-@/JO"O`KD36.:2T-_F'.
M]L>F7;=&@-@X1/"MH\88M#:^%J!&I1H"1`#&K_%?ML8[/5>#A(()?1]"6[T`
MM`&B?R.E.OK4?ARBUQ#JN=ZX;0JW1>7M:1\T'\A<!QO+5QM/!Z;U0"YV$@'I
M8FX/,+0RZT'H.$;CHO5;,[N7%A-&;+?,KPYH@A128GXC;2:=;>,0\W$O,A,1
M7<0(O]N_U6]VO.!K'W();"^-:1%2UD[!-T=@C2H'.L;\5D@[R""?/?=_P%0
M2P,$%`````@`%J.V)OI$MTE!%@``0ST```L`$`!S=')O:61S+G,N<U58#`#9
MB7XWZV9'-YP/9`"U.]MRV\:2^[@O^`?DV%OQA5((\"KK.+60,)00DP0#@):5
MAWW8G)S*5NW#V3W>JN1\#7\%("F+7DJ424J1_*1/V;X-,*`N2;8J2FQC>KI[
M9KI[^C(S^J=_?F&]L%=)%`9^;/&W&MC.*WL_4EZB;*]O>W&BL-_N!*KK,]+\
MHUW_:F<'&M9HL?PP>NI4JU7+6NQ],[X>O1Y5Z>L&OESZ^@Q?=?R:>(D'WTW\
MGDW@JXU?E[&*X'L'OL>J?Q'&T'"8Q_52];#E,$=I"5=IU;!UYB7X3:.L@N\4
M-&KN[6[-M?>.$Q7;29AX7<NZW?TX[`WLQ-OK*LN:]8.DAM-XVH#YW^YB._"Z
M0&YW@[V:;Z6^#_P`@3%?UFYW`62'>]^H_01XZJ]N$"?61'49F:D0&4`*I"A8
MG2CL,>I*)>-AM-@#9*9"9`#:^\,H4OU$**P/!9XF0<P/]V*>JW<)H7[80.U#
MAYY#T.<9`)PGFQ-J[$$8]$'CQ>HLG"[+EZD0<^QU]X==-A'?CE4^0,]+HN"=
M=1$D^X<\;2!%"H+86W8$FBADXD7A$!B\V_+>!;%UXQT!#6$B"30+`L$\9LQE
MV.T"*F`@(K;N8'['F&?A6[4*?+0(1$-T!.G%X??E`#HUWLL:P4["Z,B+_%AZ
M+@<`A^4CQI7:%ZA&0J:+*`!]>$D0]NVWP#F,8B2X'#!Z08O(S[JJ?Y`<OF[6
MGR/2213VDQR/2&`XE/N%%_5B!FLD;2N@?E]%0?_`'GB1UP-#@Q&AXRJ(-95F
MH$G>!G&P%W2#Y-BD05U>!>J("31Y0:..R";LUUIFJS!"-,#7I&0286\PU`8!
M&#Q:5TLZMB:1=Y1VNTPI/&B7`-SV0(/W$:!Q,D5!GA/EEFW2L8$OPP11A$ZX
ML*TDVFQ]Y"%K0JP3;Y_-A$A?UFB040PB!N8=Z+2?#<+N\4'8?VZ!&P3*FL^^
M)`;GEXVGY^1+6FT'I*%H%\F4GH#G.0\3]<H^"F"A>\,#.^C8CMO&-2,2,/C^
M)Z)NHR?Z;YJD_U6D8(\%8$[$+;9^U$CMJO7]S_2Y`Y[W1_V-X'_09TI@^0:P
M];?__(^_O_]I],-__0]`,ASD;ZC5$4HK'CT;1"'.5/G/&?-GP1P#GXOY;+7^
M/)I^N@3`!#W]"W:0=IP,.QW+&J?S\5FZ9A*86]5*Q^/E8GV=0VH"N<DA38%\
MSB$[UH?Y(@-0#OG>@M;%<O$QA_S5NEC,;TYF\[D&032XF`,-S0Z:#6NR3'^9
MS\ZG.:AM74V7J]GB/(=DUL4G9E!U"U%\.U1#"`O(_0)6`Z++,G!<7@*R[[.B
MM7NTKF;37\X8J9Y:UO7BY&0UI6:C5FB_8\<0257?NC'ZZRB_M9C-V>5\/5_P
MQ,8.SH7'`=)W_^92]^E,NNLH>NO%Z\=_+"N;IZ<X4M6UK$3U!FPR%DV=OCT8
MY2CDT`AS=Q#8A*'G05^!+]X/P\BW;ACL6M<N?=2M&_YHXB1@":>PAV"FR:':
M2!*VJ7L-<&K:8.+[A^!OP+PB"*X^#&Q[]I]\U8_!%?VI8A\=!A`;?'1'/9A"
M3#PI9$,\B6/<:QV``1]P6PDWA/FV30.!O(/(&*\?)O8;-8`@#<.^`8I=._*`
M**H`LYS8#OO=8UN]PRV`H=&SXQXZ(@^R'SN&"(N!!#TL4@RZWK&*>&UI]R",
M@N2P9Q]Z.%O0,4TM?@6=MO/<GG7L/=4-CVQ9)'+HZZRJKXYRB<7;2.$^MR=1
MT$D*,$)KS^WQH8+I=\+(WH?@!?XQ[%.BAOYF13HF?[%@-Q.I?=!<;*W3C)17
M;9%!@0F$D1<=Z[SGPW0](U=U<C(%8]3YW^CC:DF9CC6:3]+1$["VT6J=CL:7
MRXEEO>J&X8!Q:%(37AA#:/*:#X,@=BP]WXLLY'8]NEID9M:`446820@1*@X+
M$'HP@NA`(&-H_R_>7_<6W7K`?)(G8!3"^,@;9,..S&:)LZ$@&DK(EC"@ITX)
M%,OA:CI?C&?K&^XSPSX(Y<@+$L*:+3^=S-,/UBB;?AH1>$.*&H'9D`JLT?CL
M8O3D2^]+(1M@YN/W-?R[$GPXT/!O-3P"F^AJZ)$)C30TUM!C[RA']0U@COFO
M&ACT]]^J'/<K#?:5"7YI8*.-:_B6@6["7VOX$:F$@;8&QJ`=#8SR=7@^:^OC
M`H2?@4-_]4UX'"?!_AO!().TB+HP,M1TD;5L:I2$KW->U`Q1`1S&UOQPBJ2\
MU6(^FT#`L4;3Q7+TY&E;=%G`33(4YF@\'UNCR4DV>NK6"=0=K:9C'A>RU3*%
MZ'6#2*R@H*.NQ)FM^@([T75)CMEHE$5:/9^7@TF9ZOV*AS!*TRZ+NWWQ++
M('IC5Y088#_2W]=OH4I)L,MTPKNE+-<<;-(4.E]?I]8H/9\`1=41B\BWC-GI
MFIVX;\S.NG3RAC![VD5/5)J#7@S)FO(?<,^YP[&WMFPS^:6T,B)7],+V@Q@#
M1P5C'KE_#"<>A0#)#8\.PUCRBA@CCQT'O4&7(@;0/XOWO2YD91I;,D$,0Q:(
M"](*++/K8/98MD)DY(%%DC1A6!E$`9YJ$AX<4.D+^EWK%37KE6H%ZU\?PR@$
M*RGB>+D5[-;?V,+TA7PG<>0)8;F(>TZ:5+NP-]8;<W,CHK[7T[,+_+JAKY<.
M(Y/+O(M\L9B=KQF;/AG]9O2D`;&)K.H9<GE>N6'\9X1%S<D4'<C%'/1I09!V
M1\OU"I*8(F24C3([FXT0;6,Z7/T^.!F9C3[$8)[&)-;IM80$2CDA-,5)!1(C
ML`G*+T1R''(_5ZYO=\%DP*C(8@)(#1+[&2H`J)_O8H*C1`-<0:!Q5'7,P1(S
M#FU)ZC&^4=UQ%$:0"PW!BL((<H4$K`QLUP_[7X(9>N@L<?!3'KVO#JCV$-/\
MXKG$SB=MD+?#04^LJ'+-E)(4(_6$F'J^#UL"4RS(PXQ24H==5,VU5HTC`YQ?
MGBVRCROI0P^-AIY]?`EEU?FPM\>)-<]*-C`YPG0R'CWY,^!.TG5A7[>[QYS0
MBD:>?*TQ"%^'9VV`[`^@1]NFZ/IV][L2&XU?8K)A"I2915"D@XA5C/+H42VA
M<RGD,[[6PKMV&'+*(,II9E=M!(MOA%F#(0G=S5VZF_OH\DTA=)_OTN4@S!LK
M-^`K5$^1G./@H'\/R\*H9^<W$'_?!`-1)+I/7-SM;A7MD@U4GR/1=GE$ESP?
M*:4@6SST^@>P+U#J09\.C3J=6*'2+TY31I:ZRH`\8:\H:]2\+@<#%6UU58=L
MOP\Y=EZ:$:+FL['EY2@`P!?SM(Q90*XU+:W<)=T;JX0B8-"%<D1Y4.-(3D(.
M"6=+I?6U=EE:)0Q]"7&QZ'EI]MUH^$T):E+<Y!0B$2GOZ!2R<-HR(1K#%4IW
M(W6]=E\ZNDNL&WK_/%[,%TN7\^@.NI<!5\J,*E6TWG2,7.K2\]-6=PEAT`\Z
M'44'FU#OX!$:.>-7&%)O=[$,ZN`IV%=['N8#QC2<AP=U-@>UB%U>-*!4,*3)
M#X!M/%"@QFX*\1RU]X5E45JC]6J8N@MI31)C2+G=_>50'7TAZ4'P-O"5?>U4
MME,,_FW*#?IAU*,39L#16XK27LJ!EHLE_37Z_6!.85#+(+"*#>DQ%+=8VB;B
M7@=AC%L*R\D]E1R!Y=M;.XWM[9VZYHF;AE8%GG^G(9M1CUC=VFE"]W@UPM5:
MC"-F`EV@G*$/F]6S]V!$+1!,<<-8<K$POL.468Z%)6$P2\3D&&T<-'2PPH\'
M:C_PN@`W"CRDH2*!I>!4>2%S+U:L_/EX<0GQ&H9QF"\#\GR.,3&O%4QBJ;\E
M/:QQU8%,K(T=PILMWR$;NP8W[^UN&L?#'MXLZ$V(WJB\[Y\X.SOE??TD!\BF
M7(V7X^EYOF'U)@#+7O$.BL-NX#^Z%1'WSIZX;\%U7<JA"V3-N5)[TP*BX.`0
M5I"NYK>[[%%>N,6"EHLYNRW9;J!P8[O)`='9,>0"PX0.?2`5V?/B`/+=[C%T
M?S.,\0Q'#?(S'+NO(P:?OF+U;S\[5H.*'1^&PZYO'WJ0KT"5TAWZ"C./_)2V
M,^QS'@0NSP_I+(E2:,DU<D,X.^;;&S:&/!/!R4L'H/"=#3N$$HIT\/Z7@Q\L
M#FC7!1W[2%&"+PD?B4\?$-$Q$!X;06)7>?2T"/)6KLUHTD[#PO,960#L<+D7
M$K:653JL82T^G8RK6#&H7K`5P=X*>[G*6X:-8\8*+BU0I.4\Q:+C(-J >8V(

M;@%)\K2O>^0=QWH!(2J\.$D2BUV=I?,YB$YRI:^+]O@"(K`CML<\<CL_FTX,
M&MTBBKKXD@V*;/8A7:TTA;0$B4]T1.6P5$4:@;QE-GDDS4>B<W4TIE-2G>G=
MFQEQ-KG1428GM#^>U%"\Q`/0.U[>INCXV_H486PAW#`&0J@:Q0W=UM[N7JEN
MN$_'?YOC4-H$3(XE)WND3141I5'K]$:^I.X[\Q(L6=$V[;"X6;-XGY5M&G?;
M!P4Y'=WKW,T7\61W&X9Z816R,V9!"1\=*,Q72QV/PJ[N.)G2R0JZ,XE).![1
MXMXV!N:MI)W4$.^-[%[@0^C:BK\=PL;'D]MT.9^>Y[OUS'MGQ^K;H>KO0V2@
M.T$HW<\GN3^2.9HE,GH=XB**/(A(E<;&7L*`2N7J=6B@-PI/M,&=A/WOP)?H
MHN)<;\B:F#"S9JZ2RFH<OHS0"2872!H^7V`*MYU6$&R_QLN+K^HD4D&D;*74
MY-SXM<@($B1('>2\D,>DO"%./#DF^A>GRO]9""S-OY!.KJ'2D3292$@7"Q!N
M#@[INC%WJA6[%[X%;0$2)>D=._=MNU4X^@["N2*46687T/@\0[?&*!OCQ3P
M4C[[Z?*IN%06D`FP>QR`1:@[)Y244]X]IKS__(-V.A^6)G&Q1V>3.Z</E$9S
M9782]/&B)H+P&M/D-G;W!JVYV>^>@(O_<W1N"5&.JJQ<*G=G@D:[[^BS/\PZ
MB_(=MB'T2<H)667*=_FU:K5J#_M@'_819M7\G(/345]U]2QJ#PSF/C*8JP>[
MRZ[Q`+O:(^QJ)KMQ<3#:-;6#6S091GV^W>*ZG$+C*AN;PQ6W+]H@R$:XY-J6
M8K_(1XR3V#P!1\N?+3_I+`O:<A&A70SDS?KS9+:<ZN_369X=3Z^A'IKD/>O3
MY>6J2)V!'2=+^0V(]D154TR2F4M=3LDZXN&0.M([^@H(-$YY/5H"G5#,UAK'
M-7!.L=+8=PE%YJC1:@;:]!T4<+XBW5"Y2?,7S+UC9:"N#R-(.BV$DC"?3M-6
M9EG%C/)Q2YQ-6J-LP:LTB$8^"MWBO%U+35\@<-C%ZWF++]Y6TYE.7$BP*"RH
M/)RZ3MY+T`::GR9XVFKJ'D<KH>HX.@JXX'BR(.EY`_!VOM1%@L'TU7:.L?6U
M_;19U5D-8K4U2[S_U]5F6S.O0EV0>?TWMJ,I"(WYZM-]P&JGZ/VZD-:#_XO4
M`)*PS5(J3ZCI0%'?(K"2Y!K_=A?RY-ZPFP1T2+"]O2T[\S#`()NOO5E$P&(A
M31FOM6,D/40H)5A#*BY=:=%E8C<:A]TS;Y"GE8ML=>?:57),Z"KN4FN3G(3/
M#C16T;J6+5).1I?&T5G^],:NYMS6T_4D7>=S\+W$TZRY2[-NX\G^I,P^O_@"
MYQ[D5Q$Y;UC^4CA0:EXTB:%S'S.Z4EV:C[M*IZ<R3?U\XKYC?'D"Q\?@^HAY
M`3IX_$3?T?<1PP%F$&"^%3U,;*MD?QLV'U_^RMW@WA#/F&(ZC]GO*GW\+S;+
M+@M#2<F0<>'-8@M!"4<;!))3^VE*.T6R1P@AD',.-$U:Q0M*`)0N%/%^F\IT
M?DX4F\<Z9D%?>GNBU<'UNR0-]%8+JKQNQ#.38@C'%5^(*_2*(VE(&FY2R3)H
MA;@,+>&G3<=LN&:C9C;J9J-A-IIFHV4VVF9CQVRD9B,S&V.S,3$;4[-Q8C1:
MYGI:YGI:YGI:YGI:YGI:YGI:YGI:YGI:YGI:YGI:YGI:YGI:YGI:YGI:YGI:
MLAZJYO*#1HDM6J/W*-1E\M240&I*(#4ED)H22$T)I*8$4E,"J2F!U)1`:DH@
M-260FA)(30FDI@124P*IJ=',7$]FKB<SUY.9Z\G,]63F>C)S/9FYGLQ<3V:N
M)S/7DYGKR<SU9.9Z,G,]V;T:=46E>7@!/XDODOI0O>%)%1Z%V2'4AL:U!,=4
MQS6W<!T%)'<X3^N.V7#-1LUHM$V:MDG3-FG:0C,[-Z]WRO%8/+&4("D7.6=Y
M?"[G&VX1B]W\V+,.JQ'CG;2-T2<[9B,U&]F=>;E&VFM!;,NK;#/$+0VXV_?S
MR+>@["PE^+6\@-/M&Z,-R16"/@O(NL]/DY>G=_"CT^GUZ&3CQYK,UJ?3);[0
M<ABCT4A3\X]@N/2"GC#*_8V&]7GV`=]_U;A[.IU,LJS5TO]:GU/JKG-WJY5E
MD\ETJO]%:L``!,@@?_SAIU&6-1K3Z<F)_M<:+Q>KU>J,'NO+%*?31H/[\5\K
M6\[&_PO]+>ZO5J?T4ZUF]&.M/EVFR^F*'OF+&-+4_&/-SJ]6G^CAOW!H-,P_
MUNEB3O2.R#&;P/I:DVR2P4*FV<0Z7:V7LPO&<?0@U:KYQ[HR<')9EG]@_]$]
MD:RS_,-][@.$F.:_5W'^^MUYQ:\WH$)52>0=*C*^X`EQXXZ,2,#Q.,/O`S*
MLR6$UN^[XK[3@6^18TQ[+N@VG1X\(_-K0FVX%?B_NB5_6>]_>/^S[B"HT?&/
M4H?T(F=\"RV,3]+QU"&\FCF)MRJ"#:YD?F3S>+9@W@/RFQ&GXN+KDBN-3PQ=
M9L@X:,S\50-<IU)CG)J)(Q:IT:J`*&AU$XV,1B,Y@%:[JR7WE=U3?C#L:=&_
ML/1)\A^B&9X-3*4*$I:_<B!![@5*SX8V?DT1XEQ^HRI,T8G3V5"$B4%[?D,#
M9C\905GX*'O)_6NOY'EM+O7BS/^/D[L#JT99RE\YD"#W`J7G=\H=;?C_(71T
MU(](7+NX1X1^M8$B1L]7;&6KQ\/QWH!.U!B(KZ'C`<KN,!ALO<&#07YIW3^P
MGZD8CR<#N8Z#>$</B+KXZM^+XN>H/RD,+7V.P;\%8@P@DGU0B4VS(X3@F=RO
MQ0:J!?^2_YQ&\3_H:_.OXKF:TP2`VV2BLBELM@LB_&^GCH1N>8#-`6$DLF^8
M-\WVE1TI>K^PKXI'+6+;].#+;I:6[_`_;DFA=5ERP=H1+BZ(RVW*NWV]">J%
M\?.'MM1Z:2AQT+EM_N6O_PYK;5=:)M_!XV.W,KH?=!7`!TAHY&S8:FYDZX"
M$4CT(4.N0N?.9K>X)8VQ`R-7BP6ZOVF!#ZRLF)A3<>!_T-F#;HN[:YL8)=?G
M$$)U$Z=8H$.]3L4I%E#[+?,ON4I.CX0,!`K\W#(*Y4<:H0DH[B,B=7%.S?M1
MW!R%9NX6TZ[K:6O+*U%OA%R07/N.?S+#;1O4^BO.:0=E^VC4H'%JQ10;OU>R
M);'48<1FI?[@@'7H!9'4'Q%;G;1=V\0I>^8Z8NQ4ZL6\F[]WWIR3R]X'68(8
M&H\-V2!3=C:1KC:1'+"<QB/K:T!_&S'X7K,X=<1?K9'7$O+4M(*_$*Q_DS2V
MAS&4J0\\7Y9?LX$DN!_C;WSX=-S&OZ'I;SYFQM_A*5ZQ!C&_?\A_)0UX\6TJ
M<N3W'/K12(78TC4<SP`?:R5_Z9/6!_.+:H/A:7JK^86\HC4^C]02P,$%```
M``@`:S?;)C8,@GJC#@``9"(```L`$`!L:6(S9'8R+G)E9E58#`!$@WXW*2!V
M-YP/9`"]6FUSVL86_JS]%9M>9BZX0L;X)3B==`9C2-PAQF.[J9T/]0BQ@&HA
M44D8R*^_SSF[*PGL]KY\N)DF-F+/GK?GO*I1.#Z>N/)%I5F8Q++MM<3=\!=Y
M<GA^+D2S*<3]7,EPHGR93&6.WX\G<I;ZRWD89#(*QZF?;F68R3R1RS1YP4F9
M)0LEIWZ6RS19Y6&L,E=.DU1,DC">X0[\2Z3/"M_G?@ZVF?3CB9RD_II.+)-H
M.\-#3XBOI5B6O^49)''NAT0JX]5BK%(ZD(6S.)R&@1_G,IC[\4SQU4+%^!"H
MA8IS"!/&,O+3F9)+/\U)\FRU7";X-1G_0<8P:N*3"G++T!,]<V$8!]%JHCX(
MX23R6JWE0BT22+3PEWPWW1H&*["0J4K2F1^'W]5$?E<I;.3/E$=TMT9UZ0?!
M:H'#_,$:3,;)6N8^3.3+IR7LEJOT23B2A"7I%GZ>AAOZ-%;E#6!2AZ$7*HVV
M<I41SZ4YVF">H_N;V]OH?S<!F%X#/>RG&2SV62AK"-D0B^>`G5FOD24\-M
MK'"Y(C>37>"IGZ0?9;!>CL>9=B;Q;`9)DDXRHH0U;H97=_??AJ/#SU>R#J&F
MJXC0("=JF<^;&<Q./M>VSAK6I(4A;H:C>Y;H\K;[V_#JNB\CE<MMLG+E6D61
M*Y=1DDL6E3E:(,%M(&=3?R%U@R1*TG=2`L[0(=069M5].0]3C1/\OB@.TVTV
M+':QY_$US.&0V!\28J=A%!5BLR3XS,+!".6M+O.QG@B2-"6]V2+@0'19D"H%
M+V1+0E]*/F$UOO9O[ZY&UY8'V2$C0\"20,>"'JT1E?-J5((U3"Y$-\KGR6HV
MEXL$8;E4R3+":9(8WAS[8XT7HCPFW_LV$L*<;O%CH38!_`530*NMG"7)1"<%
MDIO/$GW*:2!5_@2ZI,P/T9[)20*``EZ>SBC=S,+82,I\$,]9F$$EV'H:OJA"
MSQE^668?2DB`./"C@!$O;`K1(`THU_@%*4G#J)2!HA#*9/WHK#D&-TH4P"HC
M-8QQ(FNXXBU"`TQ9!T2(`2'<$'?X)@UV5XY7.2>]AB`'FR`A+1<DD<EI,$X\
M42DAWK#2>&#\'A*@;!845EVOXCVR66$&'S@>^Q3EN)>^,3S\Q;9R*-:.RH1?
M@2="!\0DR`O(Z<PZ#?,<J)NFR8(0Z.?!7,-\RXSBA'41I*$B""59%HXC54GQ
M9(`K)`GD%7@O)TJ2RE(4O*D4X(LPA1V_*^8^C=0F'(=1F"/-BOM*EB?T*&-O
M.&6=I,_`+45MO)5?KWJP0/PLFTUBI>/:C];^-D/&9XNE"?0AD/H2SN(,30Q7
MV8J!O']*A;,YP+&$-928J&D8AUR?O*(6@@G9@VQE-:)"T.L.>U^Z]X[3E#V+

M309B%9T;X71[/<3^`YWK3B;5$Q9RKT_VE^^">?3<'2!WXC>(+D@JB-P#,R1
M1DV&H8.W&LE56!I4"^=F-'P<7`V'.'9)61,1C:R@)A9,.#'4[&XHE?GP63R+
ME#2UP2;EDIPP+!R3JYB[RE=I7/8*)J4*\8C4M5@!&1D2.O<,6;Z:3N5J:;/A
M0CN+H81,1O?HI!5$R9@J/E,EY'HI-#%1(B<"NEV9+0CS?ZP62Y12PBI$3G9C
MB/"2RUJGTVK!Q5^*.@YOVS]"?+.%&XZNG1PU:R?G#GQ]>P4WCBIULQXER$N%
M0QK6C;63;K-VV@9<^[\Q1K[:XFI/B-KIL?,P&@SN^OAZB&MR4W50EV=0'1],
M6:@K;^;!]RWWJ-5J@/#$>=2$=,MIL_:^[5P1"!9J$I+;7_PT).TS'<Z4Y<W]
M.H6D:D:M"H)AGB"9"&>\W;$1R`9*H6:#/8&4ZH3V#07\LU)++G`A.\H')J@D
ML9>BQ!Q]!]'.6LW:V7OY<.0^'KD/;?>QC:`ILR\[[G6MKR3!6N>B6>L,'-M/
MN*:S^'PEG(O]SL0M*JQI3D@CTY@P0[*M\H.Y<+0C+#2X"*(*F1*O2YN%+E3/
MS*$\$8[N8/PTI;80=6"Q3%:Q;1MUSRO5QE^@UI+\W>-FK=MW>JT'$AX_/E_1
MCT?ZY'F>1*CI1B\KRT_1A)K`1J7TE(<Z3WDHGY?E"9X43JJHC7QA-^D@Z)FR
M1X[>K7SU]C\SECGBKKC!$@Z:M8L6I0[`]',A3]%TFK;1H%/'%(5/K]V"=WO'
M@P'=<H$(N3AW;G>36?%-Q_GRZ_!^.#IRI?ZEK7_Y?'5D?VGOF@(7S#4W':V]
M$V9W/A@(>Y=L_HSGIZV6?=+63T[L$]RNGW3*)VV'G[RG)_51'&G<SU$"T!83
M)DW7M[2RD!7](->E`\96+[ZQ'/2]N(+5;H3SV=(3N:XY7*:E/YD`B0SE"T#Y
MHN=0XKWIWN\8FMO(I8]BG%*P<SUB:5S9V71HJ$DA@DH;=,TES-#2$?%01`1P
M)6H]F+IWHA\\%E\]4K#\#<IT&^U*SC$\0"`9[+8N*D4S>LT#UPZL=N$$)<50
M1PTES18#I-5^/]#<Y9\KM>*@Z+`OX<H!`&,[">UKB)$'GN#D[#B_[*7QHM-&
M5>;CQ.>T3[>=\FT:Q*[QG<4,9@'&=[+491&V<U9%X=';QOU@8=3DBUP3E^
M[X+JF*E.#16%\#X5?#OH&2HX:-`'U1E3G8&J=]D?&@VYTZUO&A\W!T&2U2<J
MRGUD=`(D#K\WAV_[7R1LO\"L"_,+1C"^[N#K.WSM;`Y0DDOB<QL;_"V(\:S+
MS[IX1BGQOGOA%JG2QK&K)9D<;FB$S=4,:*0Y%M073'UAJ''C6]3MT[.#JI`]
M)NJ!:#CZY*(4S%"&\OE"G\>!2SYPB0/]AQNX>K-,8@`G1`&U1]B9O3Z.7/<_
M_>6I`9\BEZ,B?ZJ8]KC-EFD?+,.#S>%1N].@/A&54\>3:31-7N&^U_H)ES&T
MI^'&M,?<IU`/C'%[D:!IUO5!&.H";!9)1*/Q89MSW:7,><C$U0G2C<#@CH0`
M%K!:2'-@M+4=SRI#VML2.^09JM>&Y2HK.EU1Z6UYM*3&A<FK7:RL=+%4J*M]
MLYGJ=`^--*S+BAX"T1D9OF@LXS=:95^8Y,;R-"H1+VTD5[HHVTMQ-$O;+?]=
MJRSNN*&C[0SZP]7R@[Q.>'<$3J,8XN3(+=)[D-ZC]+KR(\F1^T\;\W-K?G[G
MPYLP_R#W*A*I:)9([`(I>ML@HD*^BG'Z^+PE`WJ`1(2Q`_X/D,5)3=+A6-I.
M_IXF7.YW?B@7,3^\5HB2`\!;/!:.XZ?<,!`\-DU_$_(ZAC[YW&-S3'\D_!)Z
MX?X+%?@TSCL.C]7X+UL%M,#0G:ZFXIF+:@0/ZQS$F<8O+6M`.U:Z*9H4#>'.
M2!DK`B6Y$::?)(1-M:343_V^`V242[LZ-R-F4#;*Z!Z8E&GL."I`G[1%XZ[\
M])`<FO/-U$M0@<U"ZEL.8S73#4S!PZ.-B-=]A']-_:30UA;]L4@XKQ=AI=>[
ME>V8IGNWZ^@[#!%K,]BQ"D>G+1>3+('MV=/N/C/N?G3NPD5(FSUS=LM^TX?.
MS:%O^X>^FT-\JB?M#'>C4MK6P7\SS#.5R8%4_,L]!9Q0O]U9:C1>QTK9&-EV
MR#9!IO6!/4T[)M[N2/%#?_JF/Z'/)IHS-+%G?7)*M8W0?8/Q@NDI@"X$K-(K
M0IH,JL1SG_:V<Y]4A+(SE?,6D"<0&P0Z8Y83@-/38FHIM9!:1BVBD5"WV?L2
M:@?;=L?(RU%;AL-.7B'$E7OE4D&#]GK+\QZ;1\#X;GHI+Z=8TA=3;B'9673=
MU&@H#.R4XOSEB&YOJV-LSFF]U?",8CJH=3R#@1Y"FN=GGG=^BC._+GF=K`'E
MZAF:^]?)A'L53B)J=[PH"KN:Z#I4RE_TP:"C`0DJ.,Y-J>DT3+.\L*_U8&4X
M-DF08'!RY&KNL3WXLC<0ZV-=%CLN1-8+4F='9#.G54>[?=-7%M#$UVP4JFYB
M+A75><'$,[#*,/-DYC(0V^L>]#Z;[K(M-CZS44B<ITC%LWS^1-#Q:>.1JPU^
M!+A&UK,_T[R^^;TM?Y1;_O?[[^U&H^+-B+(<(B:6\.1;L5TL'?9V]=9Z'R6O
M*4[;PK;!'.X8<VY:#_*F]8B_WSB:S_^+:.YFVC`F?;EF$K;3$=W"`85@M@L:
M#86]0"[P2Z;O5+(<W1";R>(5HD4QYA13#D_-I!B-02=NX6AZQLL"UNPW/2N;
MUQ1FM<$Z_5BXW*AL<HR.4W-YF4Z&@#VO9'8F\F+(+K))`=>=;$,W[+U`26(P
M-%L?5YHM#O$\/7;QSPDS?7,71$KH?1#YQ?+;5Z?"DG2G!&OQ"S*S0I(SU-O,
M:,.OT/2]>TW61SFJJLU=P4;6[2<>S#D_/^@$_<!YKJ'+-^72Z_\EE[JO="HU
MT#GTJ"V+%29O(,WHW-Q=8^J943>L;X24%/_!4J):,O6PST_.A9WC^2-/]N*-
MF=Q]8Q@OT2NJ@S$]II$97NK#`Z%):4_<,*GLJ5QD\D5H4/DM'?5X8\JZ$XZ\
M)^YO8',$8_"\#C/UY);#S%-`ZSEDJ)C?T^69BJ:R;M95'^2)/))M>=)XC8-7
MCJ0WGA79Z\/+!_F/8YXUQAB5[`JL47KW4Y+.5+*B"2-YKKSQ)1HS5`!.VL;6
MSZ=Z.VC7T)JCL8*FP3Q7R9&['N*U(V>\EEL[(U?6=<YIV$:K4C^\-SQ*HZ/9
M%&:R>WW9O]3)L+JNX>RM(QQ"YCRJ%#J3W`6,WQFE.N6FL[H_AQ?S-2U[\W52
MI";A<#O%G91M_\U;F?UW+*;O_(E>JZW]C#:5IH]<D'0TO=&J!C-"7)0:-9V&
M`<I*L)5U6NS2<)%#.=HJ^AE&"7I'2'S>$2;^C:7U@K>T-_Z>X._9__NI>&+
MEVKFY9:&`I##,MHNE<V:3*=-LV;?:;BY;XJB9,U-XL'!]>B^?W`@KV+]`OG5
MVEK_'P3['<F!UC@[()T6_K,2TO[)5J8R59:/W,10*=6OB*.M!<U%\2:X>+=B
M7U.7#:M965>CMS3)WO&4;]'-DM>U&\Q0OYAI_KSS\AJ]?+""0#$$*N3@RJ="
M*O,$%WY/@P2&3(1"W/;$OP!02P,$%`````@`VC;;)J[1$H%;#@``_B8```D`
M$`!O8FHS9"YR96956`P`*X-^-QL?=C><#V0`U5K:;MM($GT6OZ(W$##R1)+O
MCAW#"TBT%3NP(Z_M9*V\#-I22^(,Q=:2E"^!/WY/55](2G)V%CLO*R`&V5U5
MW5W74\WT'W[?'8EYJB>IG,U4^DLF4C56J4J&*FC1+XAEEHO%?"1S]5'L;6X?
M;!X=!8\JS2*=B)WV5A`$OXH^RY')2,31`YYF:J;3%S&3\X]!\-IZ?7YMF3^5
MW^LS?J^EY^K;,CE/!$*\"F'_5'ZO;SSC#4SUO6W\V=K9VJI,U?=Z?J1^N%>:
MK8=;6W0X,+;J>T=^O)]&*LEE3L>?R3R-GD'2:=7W=SS)MT@]S764Y)Y@?[=9
MW]_S!/=Z/,Y4WA0#\R`:V3!5*A'F-1.MEGB:P@QBJ[DEHDQ@+4.Q`5G[K?J'
MTF(RC?0B$[F:S74JH75>&@9B>SQB6C[$*@OJ!UNM^L&'6NU^NSG8;M[O-`<[
MM%`631(U$ML'K8<H%T.MTU&4P-J9&.M47%_V[UC0Z4WGGY<77\Z"^F&W53\\
MJ]6N+R]N[[Y?]IO"/)U?D#B_>JY%JJ$HR/[18K&29K"1SFZKWCGS)PCO249X
M?WZ!OP-^'O`SRPZMW.M5N<&R"PP5D[2Q1*]5[Q;&O.G?777NSDO[\V*L':<B
M)S4%]2[,W3WTG%=?+^\N^]OBLK\CSB^V\6]G^92S19Q'\WAH9.56V]VCRM:Z
M48YH$(UI-)F*AY=<P9)=*+(;>I+>Q>7E=>=N:9/C*([%7.9X340CT;E9@=A/
M6_6P<"HV`JN2-<F*'+QIDTT$_>]J2-;AV2RHU1K.HN($P=!M"OF@'[%2L!PW
M;`OQKX5:0&$40;4:N088L=B-&L+68I3*IP3I(($C18EXBI!8D&28X>#0,)Q?
MT-LIV/O=R[.[6V*_U2EM2N.,.=BS'!Z_55E=4[+!Z,$6''K_U,[QJ-7^QZ#&
MTQ#[&9LBL<A0."T+%`VGCPVF.V0Z[`4O'\#T[>(6[\3T+<HBR"/9CIF)#K?!
M$9Y]8<DA.YVC21:S!Y42U2%$A?>UEKA+99+%'`@N(*RC,AF6/V>ZLF]@XHCX
M!_S$)/S8H<'O_,2#_-C%H+%^S<:).Z>S,0OL@J-8PQP^=*PL/:Q0&(\:,-VI
MH^,%3T'GR6"),[9$K]=;#C8?86P7T4A5K!$G]-*T7CDBIWQX$0A9^#2EX%;]



```
M`````0``0("!P'T``&]B:C-D+G)E9E58"``K@WXW&Q]V-U!+!08`````#0`-
+`&T#``!2C``````
`
end
```

```
.......
....
..
.                                       - fin -
```