==Phrack Inc.==

Volume 0x10, Issue 0x46, Phile #0x0a of 0x0f

```
|=-----------------------------------------------------------------------=|
|=----------------------=[ Hypervisor Necromancy; ]=---------------------=|
|=----------------=[ Reanimating Kernel Protectors, or ]=----------------=|
|=-----------------------------------------------------------------------=|
|=--------=[ On emulating hypervisors; a Samsung RKP case study ]=-------=|
|=-----------------------------------------------------------------------=|
|=---------------------------=[ Aris Thallas ]=--------------------------=|
|=--------------------=[ athallas.phrack@gmail.com ]=--------------------=|
|=-----------------------------------------------------------------------=|
```

--[Table of Contents

--[0 - Introduction

Until recently, to compromise an entire system during runtime attackers found and exploited kernel vulnerabilities. This allowed them to perform a variety of actions; executing malicious code in the context of the kernel, modify kernel data structures to elevate privileges, access protected data, etc. Various mitigations have been introduced to protect against such actions and hypervisors have also been utilized, appart from their traditional usage for virtualization support, towards this goal. In the Android ecosystem this has been facilitated by ARM virtualization extensions, which allowed vendors/OEMs to implement their own protection functionalities/logic.

On the other hand, Android devices have been universally a major PITA to debug due to the large diversity of OEMs and vendors that introduced endless customizations, the lack of public tools, debug interfaces etc. To the author's understanding, setting up a proper debug environment is usually one of the most important and time consuming tasks and can make a world of difference in understanding the under examination system or application in depth (especially true if no source code is available), identifying 0day vulnerabilities and exploiting them.

In this (rather long) article we will be investigating methods to emulate proprietary hypervisors under QEMU, which will allow researchers to interact with them in a controlled manner and debug them. Specifically, we will be presenting a minimal framework developed to bootstrap Samsung S8+ proprietary hypervisor as a demonstration, providing details and insights on key concepts on ARM low level development and virtualization extensions for interested readers to create their own frameworks and Actually Compile And Boot them ;). Finally, we will be investigating fuzzing implementations under this setup.

The article is organized as follows. The first section provides background information on ARM, Samsung hypervisors and QEMU to properly define our development setup. Next, we will elaborate on the framework implementation while dealing with the various ARM virtualization and Samsung implementation nuances. We will continue by demonstrating how to implement custom dummy fuzzers under this setup and finally for more intelligent fuzzing incorporate AFL a.k.a. "NFL or something by some chap called Cameltuft" :p

On a final note, any code snippets, memory offsets or other information presented throughout this article refer to Samsung version G955FXXU4CRJ5, QEMU version 4.1.0 and AFL version 2.56b.

--[1 - Overview

----[1.1 - ARM Architecture & Virtualization Extensions

As stated in "Arm Architecture Reference Manual Armv8, for Armv8-A architecture profile - Issue E.a" (AARM), Armv8 defines a set of Exception Levels (EL, also referred to as Execution Levels) EL0 to EL3 and two security states Secure and Non-secure aka Normal World. The higher the exception level, the higher the software execution privilege. EL3 represents the highest execution/privilege level and provides support for switching between the two security states and can access all system resources for all ELs in both security states. EL2 provides support for virtualization and in the latest version Armv8.5 support for Secure World EL2 was introduced. EL1 is the Operating System kernel EL typically described as _privileged_ and EL0 is the EL of userland applications called _unprivileged_.

```
---------------------------------------------------
|              Secure Monitor (EL3)               |
---------------------------------------------------
|   Hypervisor (EL2)*   |  Sec Hypervisor (sEL2)  |
---------------------------------------------------
|        OS (EL1)       |    Trusted OS (sEL1)    |
---------------------------------------------------
|   Userland App (EL0)  |    Secure App (sEL0)    |
---------------------------------------------------
       Normal World             Secure World
```

Switching between ELs is only allowed via taking an exception or returning from one. Taking an exception leads to a higher or the same EL while returning from one (via `eret`) to lower or the same EL. To invoke EL1, `svc` (SuperVisor Call) command is used which triggers a synchronous exception which is then handled by the corresponding OS kernel exception vector entry. Similarly, EL2 is invoked via the `hvc` (HyperVisor Call) command and EL3 via the `smc` (Secure Monitor Call) command. Switching between security states is only done by EL3.

When a hypervisor is present in the system it can control various aspects of EL1 behavior, such as trapping certain operations traditionally handled by EL1 to the hypervisor allowing the latter to decide how to handle the operation. Hypervisor Configuration Register (HCR_EL2) is the system register the allows hypervisors to define which of these behaviors they would like to enable.

Last but not least, a core feature of the virtualization extensions is the Stage 2 (S2) translation. As depicted below, this feature splits the standard translation process into two steps. First, using the EL1 translation tables (stored at Translation Table Base Register TTBRn_EL1) which are controlled by EL1, the Virtual Address (VA) is translated to an Intermediate Physical Address (IPA), instead of a Physical Address (PA) of the standard process. The IPA is then translated to a PA by the hypervisor using the Stage 2 translation table (stored at Virtual Translation Table Base Register VTTBR_EL2) which is fully controlled by EL2 and not accessible by EL1. Note that once S2 translation is enabled, EL1 does not access physical memory immediately and every IPA must always be translated via S2 tables for the actual PA access.

Of course, EL2 and EL3 maintain their own Stage 1 translation tables for their code and data VAs, which perform the traditional VA to PA mapping.

```
                                                Intermediate
  Virtual Memory Map                            Guest Physical
  Guest OS                                      Memory Map
                                                 (IPA)
 +----------------+                            +-------------+
 | +------------+ |                            | +---------+ |
 | |  OS (EL 1) | |   +--------------------+   | |  Flash  | |
 | +------------+ |   | Guest OS           |   | +---------+ |
 |                +-->+ Translation Tables +-->+             |
 | +------------+ |   | TTBRn_EL1          |   | +---------+ |
 | | APP (EL 0) | |   +--------------------+   | |   RAM   | |
 | +------------+ |                            | +---------+ |
 +----------------+                            +-------------+
                                                      |
        +---------------------------------------------+
        |
        |                                      +-------------+
        v                        Real Physical | +---------+ |
 +------+-------------+          Memory Map    | |  Flash  | |
 | Translation tables |                        | +---------+ |
 | VTTBR_EL2          +----------------------->+             |
 +--------------------+                        | +---------+ |
                                +------------->+ |   RAM   | |
                                |              | +---------+ |
 +----------------+   +---------+----------+   +-------------+
 | +------------+ |   | Hypervisor         |
 | | Hyp (EL 2) | +-->+ Translation Tables |
 | +------------+ |   | TTBR0_EL2          |
 +----------------+   +--------------------+
```

In this article we will be focusing on Normal World, implementing the EL3 and EL1 framework to bootstrap a proprietary EL2 implementation.

----[1.2 - Samsung Hypervisor

As part of its ecosystem Samsung implements a security platform named Samsung Knox [01] which among others comprises a hypervisor implementation called Real-Time Kernel Protection (RKP). RKP aims to achieve various security features [02], such as the prevention of unauthorized privileged code execution, the protection of critical kernel data (i.e. process credentials) etc.

Previous versions of the Samsung hypervisor have been targeted before, with [03] being the most notable exemplar. There, Samsung S7 hypervisor was analyzed in great detail and the article provided valuable information. Moreover, Samsung S8+ hypervisor is stripped and strings are obfuscated whereas S7 is not, providing a valuable resource for binary diffing and string comparison. Finally, the under examination S8+ hypervisor shares many similarities regarding the system architecture which have slowly begun disappearing in the latest models such as Samsung S10.

One of the most obvious differences is the location of the binary and the bootstrap process. In sum, for S8+ the hypervisor binary is embedded in the kernel image and the precompiled binary can be found in the kernel source tree under init/vmm.elf (the kernel sources are available at [04]). The kernel is also responsible for bootstrapping and initializing RKP. On the other hand, the S10+ hypervisor binary resides in a separate partition, is bootstrapped by the bootloader and then initialized by the kernel. We will provide more details in the corresponding sections that follow.

All these reasons contributed to the selection of the S8 hypervisor as the target binary, as they ease the analysis process, remove undesired complexity from secondary features/functionalities and allow focusing on the core required knowledge for our demonstration. Ultimately, though, it was an arbitrary decision and other hypervisors could have been selected.

----[1.3 - Workspace Environment

As aforementioned the targeted Samsung version is G955FXXU4CRJ5 and QEMU version is 4.1.0. Both the hypervisor and our framework are 64-bit ARM binaries. QEMU was configured to only support AArch64 targets and built with gcc version 7.4.0, while the framework was built with aarch64-linux-gnu-gcc version 8.3.0. For debugging purposes we used aarch64-eabi-linux-gdb version 7.11.

```
$ git clone git://git.qemu-project.org/qemu.git
$ cd qemu
$ git checkout v4.1.0
$ ./configure --target-list=aarch64-softmmu --enable-debug
$ make -j8
```

AFL version is 2.56b and is also compiled with gcc version 7.4.0.

```
$ git clone https://github.com/google/afl
$ cd afl
$ git checkout v2.56b
$ make
```

--[2 - Framework Implementation & RKP Analysis

The first important thing to mention regarding the framework is that it is compiled as an ELF AArch64 executable and treated as a kernel image, since QEMU allows to boot directly from ELF kernel images in EL3 and handles the image loading process. This greatly simplifies the boot process as we are not required to implement separate firmware binary to handle image loading. Function `_reset()` found in framework/boot64.S is the starting execution function and its physical address is 0x80000000 (as specified in the linker script framework/kernel.ld) instead of the default value of 0x40000000 for our QEMU setup (the reasoning behind this is explained later when the framework physical memory layout is discussed).

We are now ready to start executing and debugging the framework which is contained in the compilation output kernel.elf. We use the virt platform, cortex-a57 cpu with a single core, 3GB of RAM (the reason for this size is clarified during the memory layout discussion later), with Secure mode (EL3) and virtualization mode (EL2) enabled and wait for gdb to attach.

```
$ qemu-system-aarch64            \
    -machine virt                \
    -cpu cortex-a57              \
    -smp 1                       \
    -m 3G                        \
    -kernel kernel.elf           \
    -machine gic-version=3       \
    -machine secure=true         \
    -machine virtualization=true \
    -nographic                   \
    -S -s

$ aarch64-eabi-linux-gdb kernel.elf -q
    Reading symbols from kernel.elf...done.
    (gdb) target remote :1234
    Remote debugging using :1234
    _Reset () at boot64.S:15
    15          ldr x30, =stack_top_el3
    (gdb) disassemble
    Dump of assembler code for function _Reset:
    => 0x0000000080000000 <+0>:     ldr     x30, 0x80040000
       0x0000000080000004 <+4>:     mov     sp, x30
    ...
```

The framework boot sequence is presented below. We will explain the individual steps in the following sections. Note that we will not be following the graph in a linear manner.

```
        +-------+                  +-------+                  +-------+
        |  EL3  |                  |  EL2  |                  |  EL1  |
        +-------+                  +-------+                  +-------+
            |                          .                          .
         _reset                        .                          .
            |                          .                          .
        copy_vmm                       .                          .
            |                          .                          .
          eret  ------------------------------------------->  start_el1
            |                          .                          |
            |                          .                    __enable_mmu
            |                          .                          |
    handle_interrupt_el3  <---------------------------  smc(CINT_VMM_INIT)
            |                          .                          |
      _vmm_init_el3                    .                          |
            |                          .                          |
      eret(0xb0101000)  ---------->  start                        |
            |                          |                          |
            |                          |                          |
    handle_interrupt_el3  <--- smc(0xc2000401)                    |
            |                          |                          |
 _reset_and_drop_el1_main              |                          |
            |                          |                          |
          eret -------------------------------------------->  _el1_main
            |                          |                          |
            |                          |                       el1_main
            |                          |                          |
            |                          |                       rkp_init
            |                          |                          |
            |                          |                       rkp_call
            |                          |                          |
            |                    vmm_dispatch  <----------  hvc(RKP_INIT)
            |                          |                          |
            |               vmm_synchronous_handler               |
            |                          |                          |
            |                      rkp_main                       |
            |                          |                          |
            |                  my_handle_cmd_init                 |
            |                          |                          |
            |               various init functions...             |
            |                          |                          |
            |                  rkp_paging_init                    |
            |                          |                          |
            |                process el1 page tables              |
            |                          |                          |
            |                        eret  ----------------->  el1_main
            |                          |                          |
            |                          |                          +---+
            |                          |                          |   |
            |                          |                          |<--+
```

----[2.1 - System Bootstrap

The first thing to do after a reset is to define the stack pointers and exception vectors. Since EL2 system register values are handled by RKP during its initialization, we will be skipping EL2 registers to avoid affecting RKP configurations, except for any required reserved values as dictated by AARM. Moreover, various available oracles which will be discussed later can be examined to verify the validity of the system configuration after initializations are complete.

Stack pointers (SP_ELn) are set to predefined regions, arbitrarily sized 8kB each. Vector tables in AArch64 comprise 16 entries of 0x80 bytes each, must be 2kB aligned and are set in VBAR_ELx system configuration registers where x denotes the EL (for details refer to AARM section "D1.10 Exception entry" and "Bare-metal Boot Code for ARMv8-A Processors").

Exception taken from EL	Synchronous	IRQ	FIQ	SError
Current EL (SP_EL0)	0x000	0x080	0x100	0x180
Current EL (SP_ELx, x>0)	0x200	0x280	0x300	0x380
Lower EL AArch64	0x400	0x480	0x500	0x580
Lower EL AArch32	0x600	0x680	0x700	0x780

In our minimal implementation we will not be enabling IRQs or FIQs. Moreover, we will not be implementing any EL0 applications or performing `svc` calls from our kernel and as a result all VBAR_EL1 entries are set to lead to system hangs (infinite loops). Similarly, for EL3 we only expect synchronous exceptions from lower level AArch64 modes. As a result only the corresponding `vectors_el3` entry (+0x400) is set and all others lead to system hang as with EL1 vectors. The exception handler saves the current processor state (general purpose and state registers) and invokes the second stage handler. We follow the `smc` calling convention [05], storing the function identifier in W0 register and arguments in registers X1-X6 (even though we only use one argument). If the function identifier is unknown, then the system hangs, a decision of importance in the fuzzing setup.

```
// framework/vectors.S

.align 11
.global vectors
vectors:
    /*
     * Current EL with SP0
     */
    .align 7
    b       .           /* Synchronous */
    .align 7
    b       .           /* IRQ/vIRQ */
    ...

.align 11
.global vectors_el3
```

```
vectors_el3:
    ...

    /*
     * Lower EL, aarch64
     */
    .align 7
    b el3_synch_low_64
...

el3_synch_low_64:
    build_exception_frame

    bl  handle_interrupt_el3

    cmp x0, #0
    b.eq 1f
    b .
1:
    restore_exception_frame
    eret
...
```

Processors enter EL3 after reset and in order to drop to a lower ELs we must initialize the execution state of the desired EL and control registers and construct a fake state in the desired EL to return to via `eret`. Even though we will be dropping from EL3 directly to EL1 to allow the proprietary EL2 implementation to define its own state, we still have to set some EL2 state registers values to initialize EL1 execution state. Failure to comply with the minimal configuration results in `eret` invocation to have no effect on the executing exception level (at least in QEMU), in other words we can not drop to lower ELs.

In detail, to drop from EL3 to EL2 we have to define EL2 state in Secure Configuration Register (SCR_EL3). We set SCR_EL3.NS (bit 0) to specify that we are in Normal World, SCR_EL3.RW (bit 10) to specify that EL2 is AArch64 and any required reserved bits. Additionally, we set SCR_EL3.HCE (bit 8) to enable the `hvc` instruction here, although this could also be performed at later steps. Next, to be able to drop to EL1 we modify Hypervisor Configuration Register (HCR_EL2) to set HCR_EL2.RW (bit 31) and specify that EL1 is AArch64 and any other required reserved bits. To be as close as possible to the original setup we set some more bits here, such as HCR_EL2.SWIO (bit 1) which dictates the cache invalidation behavior. These additional values are available to us via the aforementioned oracles which will be presented later in the article.

```
// framework/boot64.S

.global _reset
_reset:
    // setup EL3 stack
    ldr x30, =stack_top_el3
    mov sp, x30

    // setup EL1 stack
    ldr x30, =stack_top_el1
    msr sp_el1, x30

    ...

    // Setup exception vectors for EL1 and EL3 (EL2 is setup by vmm)
    ldr x1, = vectors
    msr vbar_el1, x1
    ldr x1, = vectors_el3
    msr vbar_el3, x1

    ...
    // Initialize EL3 register values
    ldr   x0, =AARCH64_SCR_EL3_BOOT_VAL
    msr   scr_el3, x0

    // Initialize required EL2 register values
    mov   x0, #( AARCH64_HCR_EL2_RW )
    orr   x0, x0,#( AARCH64_HCR_EL2_SWIO )
    msr   hcr_el2, x0

    ...

    /*
     * DROP TO EL1
     */
    mov   x0, #( AARCH64_SPSR_FROM_AARCH64 | AARCH64_SPSR_MODE_EL1 | \
                 AARCH64_SPSR_SP_SEL_N)
    msr   spsr_el3, x0


    // drop to function start_el1
    adr   x0, start_el1
    msr   elr_el3, x0
    eret
```

For the fake lower level state, Exception Link Register (ELR_EL3) holds the exception return address, therefore we set it to the desired function (`start_el1()`). Saved Process Status Register (SPSR_EL3) holds the processor state (PSTATE) value before the exception, so we set its values so that the fake exception came from EL1 (SPSR_EL3.M bits[3:0]), using SP_EL1 (SPSR_EL3.M bit 0) and executing in AArch64 mode (SPSR_EL3.M bit 4). `eret` takes us to `start_el1()` in EL1. The final register related to exceptions is Exception Syndrome Register (ESR_ELx) which holds information regarding the nature of the exception (syndrome information) and as such it has no value to the returning EL and can be ignored.

------[2.1.1 - EL1

As aforementioned our goal is to provide a minimal setup. Considering this, there is also the need to be as close as possible to the original setup. Our EL1 configuration is defined with those requirements in mind and to achieve this we used system configuration register values from both the kernel source and the EL2 oracles that will be presented in the following sections, but for now we can safely assume these are arbitrarily chosen values. We will be presenting details regarding some critical system register values but for detailed descriptions please refer to AARM section "D13.2 General system control registers".

```
start_el1:
    // initialize EL1 required register values

    ldr x0, =AARCH64_TCR_EL1_BOOT_VAL
    msr tcr_el1, x0

    ldr x0, =AARCH64_SCTLR_EL1_BOOT_VAL
    msr sctlr_el1, x0
    ...
```

```
#define AARCH64_TCR_EL1_BOOT_VAL (                                \
  ( AARCH64_TCR_IPS_1TB        << AARCH64_TCR_EL1_IPS_SHIFT   ) | \
  ( AARCH64_TCR_TG1_4KB        << AARCH64_TCR_EL1_TG1_SHIFT   ) | \
  ( AARCH64_TCR_TSZ_512G       << AARCH64_TCR_EL1_T1SZ_SHIFT  ) | \
  ( AARCH64_TCR_TG0_4KB        << AARCH64_TCR_EL1_TG0_SHIFT   ) | \
  ( AARCH64_TCR_TSZ_512G       << AARCH64_TCR_EL1_T0SZ_SHIFT  ) | \
  ...
  )
```

As Translation Control Register (TCR_EL1) values suggest, we use a 40-bit 1TB sized Intermediate Physical Address space (TCR_EL1.IPS bits[34:32]), for both TTBR0_EL1 and TTBR1_EL1 4kB Translation Granule size (TCR_EL1.TG1 bits [31:30] and TCR_EL1.TG0 [15:14] respectively) and 25 size offset which means that there is a 64-25=39 bit or 512GB region of input VAs for each TTBRn_EL1 (TCR_EL1.T1SZ bits[21:16] and TCR_EL1.T0SZ bits[5:0]).

By using 4kB Granularity each translation table size is 4kB and each entry is a 64-bit descriptor, hence 512 entries per table. So at Level 3 we have 512 entries each pointing to a 4kB page or in other words we can map a 2MB space. Similarly, Level 2 has 512 entries each pointing to a 2MB space summing up to a 1GB address space and Level 1 entries point to 1GB spaces summing up to a 512GB address space. In this setup where there are 39bit input VAs we do not require a Level 0 table as shown from the translation graph. For more details refer to AARM section "D5.2 The VMSAv8-64 address translation system".

```
+---------+---------+---------+-----------+
| [38:30] | [29:21] | [20:12] |  [11:0]   | VA segmentation with
|         |         |         |           | 4kB Translation Granule
| Level 1 | Level 2 | Level 3 | Block off | 512GB input address space
+---------+---------+---------+-----------+
```

```
                                         Physical Address
                             +-------------------------+-----------+
 VA Translation              |          [39:12]        |  [11:0]   |
 demonstration with          +-------------------------+-----------+
 4kB Granule,                               ^                 ^
 512GB Input VA Space                       |                 |
 1TB IPS                                    |                 +-----------+
                                            +-------------------------+  |
                                                                      |  |
                Level 1 tlb          Level 2 tlb        Level 3 tlb   |  |
    +--------> +-----------+   +--->+-----------+  +-->+-----------+  |  |
    |          |           |   |    |           |  |   |           |  |  |
    |          +-----------+   |    +-----------+  |   |           |  |  |
    |          | 1GB block |   |    | 2MB block |  |   |           |  |  |
    |          |   entry   |   |    |   entry   |  |   |           |  |  |
    |          +-----------+   |    +-----------+  |   |           |  |  |
    |          |           |   |    |           |  |   |           |  |  |
    |          +-----------+   |    |           |  |   |           |  |  |
    |      +-->+ Tbl entry +---+    |           |  |   |           |  |  |
+---+---+  |   +-----------+        +-----------+  |   |           |  |  |
| TTBRn |  |   |           |    +-->+ Tbl entry +--+   +-----------+  |  |
+---+---+  |   |           |    |   +-----------+   +->+ Pg entry  +--+  |
    ^      |   |           |    |   |           |   |  +-----------+     |
    |      |   |           |    |   |           |   |  |           |     |
    +--+   |   +-----------+    |   +-----------+   |  +-----------+     |
       |   |                    +------+            |                    |
       |   +----+ Index +----+         |         +--+        +-----------+
       |                     |         |         |           |
+----+-+-+----+---------+----+----+----+----+----+----+------+----+
|    |   |    | Level 0 | Level 1 | Level 2 | Level 3 | PA offset | VA
+----+---+----+---------+---------+---------+---------+-----------+
      [55]      [47:39]   [38:30]   [29:21]   [20:12]     [11:0]
  TTBRn Select
```

For Levels 1 and 2 every entry can either point to the next translation table level (table entry) or to the actual physical address (block entry) effectively ending translation. The entry type is defined in bits[1:0], where bit 0 identifies whether the descriptor is valid (1 denotes a valid descriptor) and bit 1 identifies the type, value 0 being used for block entries and 1 for table entries. As a result entry type value 3 identifies table entries and value 1 block entries. Level 1 block entries point to 1GB memory regions with VA bits[29:0] being used as the PA offset and Level 2 block entries point to 2MB regions with bits[20:0] used as the offset. Last but not least, Level 3 translation tables can only have page entries (similar to block entries but with descriptor type value 3, as previous level table entries).

```
61          51                             11         2  1:0
+------------+-----------------------------+----------+------+ Block Entry
| Upper Attr |           ...               | Low Attr | Type | Stage 1
+------------+-----------------------------+----------+------+ Translation

| bits | Attr      | Description                    |
---------------------------------------------------
| 4:2  | AttrIndex | MAIR_EL1 index                 |
| 7:6  | AP        | Access permissions             |
| 53   | PXN       | Privileged execute never       |
| 54   | (U)XN     | (Unprivileged) execute never   |
                                                    Block entry attributes
| AP | EL0 Access   | EL1/2/3 Access |              for Stage 1 translation
-------------------------------------
| 00 | None         |   Read Write   |
| 01 | Read Write   |   Read Write   |
| 10 | None         |   Read Only    |
| 11 | Read Only    |   Read Only    |
```

```
61      59                                          2  1:0
+--------+--------------------------------------------+------+ Table Entry
|  Attr  |                    ...                     | Type | Stage 1
+--------+--------------------------------------------+------+ Translation

| bits  | Attr | Description                 |
---------------------------------------------
| 59    | PXN  | Privileged execute never    |
| 60    | U/XN | Unprivileged execute never  |
| 62:61 | AP   | Access permissions          |
                                                Table entry attributes
| AP | Effect in subsequent lookup levels |     for Stage 1 translation
-------------------------------------------
| 00 | No effect                          |
| 01 | EL0 access not permitted           |
| 10 | Write disabled                     |
| 11 | Write disabled, EL0 Read disabled  |
```

In our setup we use 2MB regions to map the kernel and create two mappings. Firstly, an identity mapping (VAs are equal to the PAs they are mapped to) set to TTBR0_EL1 and used mainly when the system transitions from not using the MMU to enabling it. Secondly, the TTBR1_EL1 mapping where PAs are mapped to VA_OFFSET + PA, which means that getting the PA from a TTBR1_EL1 VA or vice versa is simply done by subtracting or adding the VA_OFFSET correspondingly. This will be of importance during the RKP initialization.

```
#define VA_OFFSET 0xffffff8000000000

#define __pa(x) ((uint64_t)x - VA_OFFSET)
#define __va(x) ((uint64_t)x + VA_OFFSET)
```

The code to create the page tables and enable the MMU borrows heavily from the Linux kernel implementation. We use one Level 1 entry and the required amount of Level 2 block entries with the two tables residing in contiguous preallocated (defined in the linker script) physical pages. The Level 1 entry is evaluated by macro `create_table_entry`. First, the entry index is extracted from VA bits[38:30]. The entry value is the next Level table PA

ORed with the valid table entry value. This also implicitly defines the table entry attributes, where (U)XN is disabled, Access Permissions (AP) have no effect in subsequent levels of lookup. For additional details regarding the memory attributes and their hierarchical control over memory accesses refer to AARM section "D5.3.3 Memory attribute fields in the VMSAv8-64 translation table format descriptors".

A similar process is followed for Level 2 but in a loop to map all required VAs in macro `create_block_map`. The entry value is the PA we want to map ORed with block entry attribute values defined by AARCH64_BLOCK_DEF_FLAGS. The flag value used denotes a non-secure memory region, (U/P)XN disabled, Normal memory as defined in Memory Attribute Indirection Register (MAIR_EL1) and Access Permissions (AP) that allow Read/Write to EL1 and no access to EL0. As with table entries, for detailed description refer to AARM section "D5.3.3". Finally, MAIR_ELx serves as a table holding information/attributes of memory regions and readers may refer to AARM section "B2.7 Memory types and attributes" for more information.

```
// framework/aarch64.h

/*
 * Block default flags for initial MMU setup
 *
 * block entry
 * attr index 4
 * NS = 0
 * AP = 0 (EL0 no access, EL1 rw)
 * (U/P)XN disabled
 */
#define AARCH64_BLOCK_DEF_FLAGS (                                 \
  AARCH64_PGTBL_BLK_ENTRY                                 | \
  0x4 << AARCH64_PGTBL_BLK_ENT_STAGE1_LOW_ATTR_IDX_SHIFT | \
  AARCH64_PGTBL_BLK_ENT_STAGE1_LOW_ATTR_AP_RW_ELHIGH <<     \
      AARCH64_PGTBL_BLK_ENT_STAGE1_LOW_ATTR_AP_SHIFT     | \
  AARCH64_PGTBL_BLK_ENT_STAGE1_LOW_ATTR_SH_INN_SH    <<     \
      AARCH64_PGTBL_BLK_ENT_STAGE1_LOW_ATTR_SH_SHIFT     | \
  1 << AARCH64_PGTBL_BLK_ENT_STAGE1_LOW_ATTR_AF_SHIFT      \
  )
```

```
// framework/mmu.S

__enable_mmu:
    ...

    bl __create_page_tables

    isb
    mrs x0, sctlr_el1
    orr x0, x0, #(AARCH64_SCTLR_EL1_M)
    msr sctlr_el1, x0
...

__create_page_tables:

    mov x7, AARCH64_BLOCK_DEF_FLAGS
    ...

    // x25 = swapper_pg_dir
    u/ x20 = VA_OFFSET
    mov x0, x25
    adrp x1, _text
    add x1, x1, x20
    create_table_entry x0, x1, #(LEVEL1_4K_INDEX_SHIFT), \
                        #(PGTBL_ENTRIES), x4, x5

    adrp x1, _text
    add x2, x20, x1
    adrp x3, _etext
    add x3, x3, x20
    create_block_map x0, x7, x1, x2, x3
...

.macro create_table_entry, tbl, virt, shift, ptrs, tmp1, tmp2
    lsr \tmp1, \virt, \shift
    and \tmp1, \tmp1, \ptrs - 1                 // table entry index
    add \tmp2, \tbl, #PAGE_SIZE                 // next page table PA
    orr \tmp2, \tmp2, #AARCH64_PGTBL_TBL_ENTRY // valid table entry
    str \tmp2, [\tbl, \tmp1, lsl #3]            // store new entry
    add \tbl, \tbl, #PAGE_SIZE                  // next level table page
.endm

.macro  create_block_map, tbl, flags, phys, start, end
    lsr     \phys, \phys, #LEVEL2_4K_INDEX_SHIFT
    lsr     \start, \start, #LEVEL2_4K_INDEX_SHIFT
    and     \start, \start, #LEVEL_4K_INDEX_MASK     // table index
    orr     \phys, \flags, \phys, lsl #LEVEL2_4K_INDEX_SHIFT // table entry
    lsr     \end, \end, #LEVEL2_4K_INDEX_SHIFT     // block entries counter
    and     \end, \end, #LEVEL_4K_INDEX_MASK       // table end index
    1:   str     \phys, [\tbl, \start, lsl #3]  // store the entry
    add     \start, \start, #1                     // next entry
    add     \phys, \phys, #LEVEL2_4K_BLK_SIZE      // next block
    cmp     \start, \end
    b.ls    1b
.endm

...
```

As a demonstration we perform a manual table walk for VA 0xffffff8080000000 which should be the TTBR1_EL1 VA of function `_reset()`. The Level 1 table index (1) is 2 and the entry value is 0x8008a003 which denotes a valid table descriptor at PA 0x8008a000. The Level 2 entry index (2) is 0 and value of the entry is 0x80000711 which denotes a block entry at physical address 0x80000000. The remaining VA bits setting the PA offset are zero and examining the resulting PA is of course the start of function `_reset()`. Note that since we have not yet enabled the MMU (as shown in the disassembly this is performed in the next instructions), all memory accesses with gdb refer to PAs that is why we can directly examine the page tables and resulting PA. In our setup that would be true even with MMU enabled due to the identity mapping, however, this should not be assumed to apply to every system.

```
(gdb) disas
Dump of assembler code for function __enable_mmu:
   0x00000000800401a0 <+0>:     mov     x28, x30
   0x00000000800401a4 <+4>:     adrp    x25, 0x80089000 // TTBR1_EL1
   0x00000000800401a8 <+8>:     adrp    x26, 0x8008c000
   0x00000000800401ac <+12>:    bl      0x80040058 <__create_page_tables>
=> 0x00000000800401b0 <+16>:    isb
   0x00000000800401b4 <+20>:    mrs     x0, sctlr_el1
   0x00000000800401b8 <+24>:    orr     x0, x0, #0x1
End of assembler dump.

(gdb)  p/x ((0xffffff8000000000 + 0x80000000) >> 30) & 0x1ff      /* (1) */
$19 = 0x2

(gdb)  x/gx ($TTBR1_EL1 + 2*8)
0x80089010:     0x000000008008a003

(gdb)  p/x ((0xffffff8000000000 + 0x80000000) >> 21) & 0x1ff      /* (2) */
$20 = 0x0

(gdb)  x/gx 0x000000008008a000
0x8008a000:     0x0000000080000711
```

```
(gdb)  x/10i 0x0000000080000000
   0x80000000 <_reset>: ldr     x30, 0x80040000
   0x80000004 <_reset+4>:       mov     sp, x30
   0x80000008 <_reset+8>:       mrs     x0, currentel
```

Finally, with the MMU enabled we are ready to enable RKP. Since the EL2 exception vector tables are not set, the only way to do that is to drop to EL2 from EL3 as we did for EL1. We invoke `smc` with function identifier CINT_VMM_INIT which the EL3 interrupt handler redirects to function `_vmm_init_el3()`.

----[2.2 - EL2 Bootstrap

RKP binary is embedded in our kernel image using the `incbin` assembler directive as shown below and before dropping to EL2 we must place the binary in its expected physical address. Since RKP is an ELF file, we can easily obtain the PA and entry point which for this specific RKP version are 0xb0100000 and 0xb0101000 respectively. `copy_vmm()` function copies the binary from its kernel position to the expected PA during the system initialization in function `_reset()`.

```
// framework/boot64.S
...

.global _svmm
_svmm:
.incbin "vmm-G955FXXU4CRJ5.elf"
.global _evmm
_evmm:
...
```

```
$ readelf -l vmm-G955FXXU4CRJ5.elf

Elf file type is EXEC (Executable file)
Entry point 0xb0101000
There are 2 program headers, starting at offset 64

Program Headers:
  Type           Offset             VirtAddr           PhysAddr
                 FileSiz            MemSiz              Flags  Align
  LOAD           0x0000000000000000 0x00000000b0100000 0x00000000b0100000
                 0x000000000003e2e0 0x000000000003e6c0  RWE    0x10000
  ...
```

At long last we are ready to drop to EL2. Similarly to dropping to EL1, we set ELR_EL3 to the RKP entry point and SPSR_EL3 so that the fake exception came from EL2 executing in AArch64 mode. We additionally set X0 and X1 to to RKP start PA and reserved size. These values are dictated by the Samsung kernel implementation and the oracles and required by the EL2 implementation which will be explained shortly. Readers interested in the Samsung kernel implementation can refer to kernel function `vmm_init()` at kernel/init/vmm.c which is called during the kernel initialization in function `start_kernel()`.

```
// framework/boot64.S

.global _vmm_init_el3
.align 2
_vmm_init_el3:
    // return to vmm.elf entry (RKP_VMM_START + 0x1000)
    mov   x0, #RKP_VMM_START
    add   x0, x0, #0x1000
    msr   elr_el3, x0
    mov   x0, #(AARCH64_SPSR_FROM_AARCH64 | AARCH64_SPSR_MODE_EL2 | \
                AARCH64_SPSR_SP_SEL_N)
    msr   spsr_el3, x0

    // these are required for the correct hypervisor setup
    mov   x0, #RKP_VMM_START
    mov   x1, #RKP_VMM_SIZE
    eret
    .inst       0xdeadc0de //crash for sure
ENDPROC(_vmm_init_el3)
```

One valuable source of information at this point is the Linux kernel procfs entry /proc/sec_log as it provides information about the aforementioned values during Samsung kernel `vmm_init()` invocation. This procfs entry is part of the Exynos-SnapShot debugging framework and more information can be found in the kernel source at kernel/drivers/trace/exynos-ss.c. A sample output with RKP related values is displayed below. Apart from the RKP related values we can see the kernel memory layout which will be helpful in creating our framework memory layout to satisfy the plethora of criteria introduced by RKP which will be presented later.

```
RKP: rkp_reserve_mem, base:0xaf400000, size:0x600000
RKP: rkp_reserve_mem, base:0xafc00000, size:0x500000
RKP: rkp_reserve_mem, base:0xb0100000, size:0x100000
RKP: rkp_reserve_mem, base:0xb0200000, size:0x40000
RKP: rkp_reserve_mem, base:0xb0400000, size:0x7000
RKP: rkp_reserve_mem, base:0xb0407000, size:0x1000
RKP: rkp_reserve_mem, base:0xb0408000, size:0x7f8000
software IO TLB [mem 0x8f9680000-0x8f9a80000] (4MB) mapped at
    [ffffffc879680000-ffffffc879a7ffff] Memory: 3343540K/4136960K available
    (11496K kernel code, 3529K rwdata, 7424K rodata, 6360K init, 8406K bss,
    637772K reserved, 155648K cma-reserved)
Virtual kernel memory layout:
    modules : 0xffffff8000000000 - 0xffffff8008000000   (   128 MB)
    vmalloc : 0xffffff8008000000 - 0xffffffbdbfff0000   (   246 GB)
      .init : 0xffffff8009373000 - 0xffffff80099a9000   (  6360 KB)
      .text : 0xffffff80080f4000 - 0xffffff8008c2f000   ( 11500 KB)
    .rodata : 0xffffff8008c2f000 - 0xffffff8009373000   (  7440 KB)
      .data : 0xffffff80099a9000 - 0xffffff8009d1b5d8   (  3530 KB)
    vmemmap : 0xffffffbdc0000000 - 0xffffffbfc0000000   (     8 GB maximum)
              0xffffffbdc0000000 - 0xffffffbde2000000   (   544 MB actual)
SLUB: HWalign=64, Order=0-3, MinObjects=0, CPUs=8, Nodes=1
RKP: vmm_reserved
  .base=ffffffc030100000 .size=1048576
  .bss=ffffffc03013e2e0 .bss_size=992
  .text_head=ffffffc030101000 .text_head_size=192
RKP: vmm_kimage
.base=ffffff8009375a10  .size=255184
RKP: vmm_start=b0100000, vmm_size=1048576
RKP: entry point=00000000b0101000
RKP: status=0
in rkp_init, swapper_pg_dir : ffffff800a554000
```

The entry point eventually leads to RKP function `vmm_main()` (0xb0101818). The function initially checks whether RKP has already been initialized (3) and if true it returns, or else proceeds with the initialization and sets the initialization flag. Immediately after this, `memory_init()` function (0xb0101f24) is called where a flag is set indicating that memory is active and a 0x1f000 sized buffer at 0xb0220000 is initialized to zero.

```
// vmm-G955FXXU4CRJ5.elf

int64_t vmm_main(int64_t hyp_base_arg, int64_t hyp_size_arg, char **stacks)
{
    ...
    if ( !initialized_ptr )                                       /* (3) */
    {
        initialized_ptr = 1;
        memory_init();

        log_message("RKP_cdb5900c %sRKP_b826bc5a %s\n",
                                    "Jul 11 2018", "11:19:43");

        /* various log messages and misc initializations */

        heap_init(base, size);
        stacks = memalign(8, 0x10000) + 0x2000;

        vmm_init();
        ...

        if (hyp_base_arg != 0xB0100000)
            return -1;
        ...

        set_ttbr0_el2(&_static_s1_page_tables_start___ptr);
        s1_enable();

        set_vttbr_el2(&_static_s2_page_tables_start___ptr);
        s2_enable();
    }
    ...

    return result;
}
```

This buffer is the RKP log and along with RKP debug log at 0xb0200000, which will be presented later, they comprise the EL2 oracles. Both of them are made available via procfs entry /proc/rkp_log and interested readers can check kernel/drivers/rkp/rkp_debug_log.c for more information from the kernel perspective. RKP log is written to by `log_message()` function (0xb0102e94) among others and an edited sample output from `vmm_main()` with deobfuscated strings as comments with the help of S7 hypervisor binary as mentioned before.

```
RKP_1f22e931 0xb0100000 RKP_dd15365a 40880   // file base: %p size %s
RKP_be7bb431 0xb0100000 RKP_dd15365a 100000  // region base: %p size %s
RKP_2db69dc3 0xb0220000 RKP_dd15365a 1f000   // memory log base: %p size %s
RKP_2c60d5a7 0xb0141000 RKP_dd15365a bf000   // heap base: %p size %s
```

During the initialization the heap is initialized and memory is allocated for the stack which has been temporarily set to a reserved region during compilation. Next, in `vmm_init()` (0xb0109758) two critical actions are performed. First, the EL2 exception vector table (0xb010b800) is set in VBAR_EL2 enabling us to invoke RKP from EL1 via `hvc`. Finally, HCR_EL2.TVM (bit 26) is set trapping EL1 writes to virtual memory control registers (SCTLR_EL1, TTBRnRL1, TCR_EL1, etc) to EL2 with Syndrome Exception Class (ESR_EL2.EC bits [31:26]) value 0x18 (more on this while discussing the EL2 synchronous exception handler).

At this point we clarify one the aforementioned constrains; that of the RKP bootstrap arguments. The RKP PA is compared at this point with hardcoded value 0xb0100000 and if there's a mismatch the bootstrap process terminates and -1 is returned denoting failure. Furthermore, the PA is stored and used later during the paging initialization, also discussed later.

If the RKP PA check is satisfied, the final bootstrap steps comprise the MMU and memory translations enabling. First, EL2 Stage 1 translations are enabled. TTBR0_EL2 is set to predefined static tables at 0xb011a000 and `s1_enable()` (0xb0103dcc) function is called. First, MAIR_EL2 is set to define two memory attributes (one for normal memory and one for device memory). Next, TCR_EL2 is ORed with 0x23518 which defines a 40 bits or 1TB Physical Address Size (TCR_EL2.PS bits[18:16]), a 4kB Granule size (TCR_EL2.TG0 bits[15:14]) and 24 size offset (TCR_EL2.T0SZ bits[5:0]) which corresponds to a 64-24=40 bit or 1TB input address space for TTBR0_EL2. To conclude `s1_enable()` SCTLR_EL2 is set with the important values being SCTLR_EL2.WNX (bit 19) which enables the behavior where write permission implies XN and SCTLR_EL2.M (bit 0) which enables the MMU.

Last but not least, Stage 2 translation is enabled. VTTBR_EL2 which holds the Stage 2 translation tables is set to the predefined static tables at 0xb012a000. Next, Virtual Translation Control Register (VTCR_EL2) is set which as the name dictates, controls the Stage 2 translation process similarly to TCR_ELx for Stage 1 translations. Its value defines a 40 bits or 1TB Physical Address Size (VTCR_EL2.PS bits[18:16]), a 4kB Granule size (TCR_EL2.TG0 bits[15:14]), and 24 size offset (TCR_EL2.T0SZ bits[5:0]) which corresponds to a 64-24=40 bit or 1TB input address space for VTTBR0_EL2. Moreover, Starting Level of Stage 2 translation controlled by VTCR_EL2.SL0 (bits[7:6]) is set to 1 and since TCR_EL2.TG0 is set to 4kB Stage 2 translations start at Level 1 with concatenated tables which will be explained in detail next. Finally, HCR_EL2.VM (bit 0) is set to enable Stage 2 translation.

------[2.2.1 - Stage 2 translation & Concatenated tables

As AARM states "for a Stage 2 translation, up to 16 translation tables can be concatenated at the initial lookup level. For certain input address sizes, concatenating tables in this way means that the lookup starts at a lower level than would otherwise be the case". We are going to demonstrate this in our current setup but for more details refer to section "D5.2.6 Overview of the VMSAv8-64 address translation stages" of AARM.

Since we have a 40 bit input address range only bit 39 of the input VA is used to index translation table at Level 0 and as a result only two Level 1 tables exist. Instead of the default setup, ARM allows to concatenate the two tables in contiguous physical pages and start translation in Level 1. To index the Level 1 tables, IPA bits[39:30] are used instead of the traditional bits[38:30].

```
+---------+---------+---------+---------+-----------+ Default approach
|   39    | [38:30] | [29:21] | [20:12] |  [11:0]   | Stage 2 translation
|         |         |         |         |           | IPA segmentation
| Level 0 | Level 1 | Level 2 | Level 3 | Block off | 4kB Granule
+---------+---------+---------+---------+-----------+ 40-bit IPS

+-------------+---------+---------+-----------+ Concatenated Tables
|   [39:30]   | [29:21] | [20:12] |  [11:0]   | IPA segmentation
|             |         |         |           | 4kB Granule
|   Level 1   | Level 2 | Level 3 | Block off | 40-bit IPS
+-------------+---------+---------+-----------+ VTCR_EL2.SL0 = 1
```

We have included a gdb script to dump the Stage 2 translation tables based on tools from [03] and [06]. The script reads the table PA from VTTBR_EL2 and is configured for our setup only and not the generic approach. Moreover, it needs to be called from EL2 or EL3, for which `switchel <#>` command can be used. Finally, our analysis indicates that there is a 1:1 mapping between IPAs and PAs.

```
(gdb) switchel
$cpsr = 0x5 (EL1)

(gdb)  switchel 2
Moving to EL2
$cpsr = 0x9

(gdb) pagewalk
```

```
################################################
#      Dump Second Stage Translation Tables     #
################################################
PA Size: 40-bits
Starting Level: 1
IPA range: 0x000000ffffffffff
Page Size: 4KB

...
Third level: 0x1c07d000-0x1c07e000: S2AP=11, XN=10
Third level: 0x1c07e000-0x1c07f000: S2AP=11, XN=10
...
second level block: 0xbfc00000-0xbfe00000: S2AP=11, XN=0
second level block: 0xbfe00000-0xc0000000: S2AP=11, XN=0
first level block: 0xc0000000-0x100000000: S2AP=11, XN=0
first level block: 0x880000000-0x8c0000000: S2AP=11, XN=0
...

(gdb) switchel 1
Moving to EL1
$cpsr = 0x5 (EL1)
```

------[2.2.2 - EL2 bootstrap termination and EL1 physical address

Now that the hypervisor is setup we can resume with the framework setup. The bootstrap process terminates via an `smc` command thus returning to EL3. X0 holds the special value 0xc2000401 and X1 the return value of the operation (zero denoting success). If the bootstrap process fails, `handle_interrupt_el3()` fails (5) and the system hangs (4).

```
// framework/vectors.S

el3_synch_low_64:
    build_exception_frame

    bl  handle_interrupt_el3

    cmp x0, #0                                                  /* (4) */
    b.eq 1f
    b .
1:
    restore_exception_frame
    eret
...

// framework/interrupt-handler.c

int handle_interrupt_el3(uint64_t value, uint64_t status)
{
    int ret = 0;
    switch (value) {
        case 0xc2000401: // special return value from vmm initialization
            if (status == 0) {
                _reset_and_drop_el1_main();
            } else {
                ret = -1;                                       /* (5) */
            }
     ...
}
```

Careful readers might have noticed that the EL2 `smc` invocation causes a new exception frame to be stored in EL3 and in order to return to EL1 we must properly restore the state. Well, due to the framework minimal nature no information needs to be saved before or after EL2 bootstrap. As a result we simply reset the state (i.e. stack pointers) and drop to EL1 function `_el1_main()` which in turn leads to `el1_main()`.

```
// framework/boot64.S

...

_reset_and_drop_el1_main:
    /*
     * We have initialized vmm. Jump to EL1 main since HVC is now enabled,
     * and EL1 does not require EL3 to interact with hypervisor
     */
    // setup EL3 stack
    ldr x30, =stack_top_el3
    mov sp, x30

    // setup EL1 stack
    ldr x30, =stack_top_el1
    msr sp_el1, x30

    mov   x0, #(AARCH64_SPSR_FROM_AARCH64 | AARCH64_SPSR_MODE_EL1 | \
                AARCH64_SPSR_SP_SEL_N)
    msr   spsr_el3, x0


    // drop to function _el1_main
    adr   x0, _el1_main
    msr   elr_el3, x0
    eret                                                          /* (6) */
...

_el1_main:
    mov x20, #-1
    lsl x20, x20, #VA_BITS
    adr x0, el1_main
    add x0, x0, x20

    blr x0
...
```

Here we explain another system constrain. Our framework was arbitrarily placed at PA 0x80000000. The reason should by now be obvious. After enabling Stage 2 translation, every EL1 IPA is translated through Stage 2 tables to find the PA. Examining the hypervisor static maps reveals region starting at 0x80000000 to satisfy the criteria required for lower level execution. Specifically, eXecute Never (XN) field is unset and there is no write permissions. Should the kernel be placed in an unmapped or non executable for Stage 2 translation region during framework initialization, then returning from EL3 to EL1 (6) results in a translation error.

```
(gdb) pagewalk

################################################
#     Dump Second Stage Translation Tables     #
################################################
...
Third level: 0x1c07e000-0x1c07f000: S2AP=11, XN=10
Third level: 0x1c07f000-0x1c080000: S2AP=11, XN=10
Third level: 0x80000000-0x80001000: S2AP=1, XN=0
Third level: 0x80001000-0x80002000: S2AP=1, XN=0
...
```

```
54          51                             10         2  1:0
+------------+-----------------------------+----------+------+ Block Entry
| Upper Attr |           ....              | Low Attr | Type | Stage 2
+------------+-----------------------------+----------+------+ Translation
```

bits	Attr	Description
5:2	AttrIndex	MAIR_EL2 index
7:6	S2AP	Access permissions
53:54	XN	Execute never

S2AP	EL1/EL0 Access
00	None
01	Read Only
10	Write Only
11	Read Write

XN	Allow Exec
00	EL0/EL1
01	EL0 not EL1
10	None
11	EL1 not EL0

Block entry attributes
for Stage 2 translation

----[2.3 - RKP Initialization Functions

The first thing performed in `el1_main()` is to initialize RKP. There are numerous steps that comprise RKP initialization and we will present them in the following sections. Before explaining the initialization process though we will describe the RKP exception handlers.

------[2.3.1 - RKP Synchronous Handler

As explained during the EL2 bootstrap VBAR_EL2 is set at 0xb010b800 where each handler first creates the exception frame storing all generic registers and then calls function `vmm_dispatch()` (0x0b010aa44) with the three arguments being the offset indicating the EL from which the exception was taken, the exception type and the exception frame address respectively. `vmm_dispatch()` is designed to only handle synchronous exceptions and simply returns otherwise. Function `vmm_synchronous_handler()` (0xb010a678) handles as the name suggests the synchronous exceptions and only the exception frame (third) argument is of importance.

```
stp    X1, X0, [SP,#exception_frame]!
...
mov    X0, #0x400       // Lower AArch64
mov    X1, #0           // Synchronous Exception
mov    X2, SP           // Exception frame, holding args from EL1

bl     vmm_dispatch
...
ldp    X1, X0, [SP+0x10+exception_frame],#0x10
clrex
eret
```

As shown from the following snippet the handler first evaluates ESR_EL2.EC. Data and Instruction Aborts from the current EL (ECs 0x21 and 0x25) are not recoverable and the handler calls `vmm_panic()` function (0xb010a4cc) which leads to system hang. Data and Instruction Aborts from lower EL (ECs 0x20 and 0x24) are handled directly by the handler. Furthermore, as mentioned before, by setting HCR_EL2.TVM during the RKP bootstrap, EL1 writes to virtual memory control registers are trapped to EL2 with EC 0x18 and here handled by function `other_msr_mrs_system()` (0xb010a24c). `hvc` commands either from AArch32 or AArch64 (ECs 0x12 and 0x16) are our main focus and will be explained shortly. Finally, any other ECs return -1 which leads `vmm_dispatch()` to `vmm_panic()`.

```
// vmm-G955FXXU4CRJ5.elf

int64_t vmm_synchronous_handler(int64_t from_el_offset,
                int64_t exception_type, exception_frame *exception_frame) {
```

```
    esr_el2 = get_esr_el2();
    ...

    switch ( esr_el2 >> 26 )   /* Exception Class */
    {
        case 0x12:        /* HVC from AArch32 */
        case 0x16:        /* HVC from AArch64 */


            if ((exception_frame->x0 & 0xFFF00000) == 0x83800000) /* (7) */
                rkp_main(exception_frame->x0, exception_frame);
            ...
            return 0;


        case 0x18:   /* Trapped MSR, MRS or System instruction execution */
            v7 = other_msr_mrs_system(exception_frame);
            ...

        case 0x20:     /* Instruction Abort from a lower Exception level */
            ...

        case 0x21:          /* Instruction Abort Current Exception Level */
            vmm_panic(from_el_offset, exception_type, ...);

        case 0x24:            /* Data Abort from a lower Exception level */
            ...

        case 0x25:                 /* Data Abort Current Exception Level */
            vmm_panic(from_el_offset, exception_type, ...);

        default:
            return -1;
    }
}
```

Before moving to `hvc` we will be briefly introducing `msr`/`mrs` handling (for details regarding the values of ESR_EL2 discussed here refer to AARM section "D13.2.37"). First, the operation direction is checked via the ESR_EL2.ISS bit 0. As mentioned only writes are supposed to be trapped (direction bit value must be 0) and if somehow a read was trapped, handler ends up in `vmm_panic()`. The general purpose register used for the transfer is discovered from the value of ESR_EL2.ISS.Rt (bits [9:5]). The rest of ESR_EL2.ISS values are used to identify the system register accessed by `msr` and in RKP each system register is handled differently. For example SCTLR_EL1 handler does not allow to disable the MMU or change endianess and TCR_EL1 handler does not allow modification of the Granule size. We will not be examining every case in this (already long) article, but interested readers should by now have more than enough information to start investigating function `other_msr_mrs_system()`.

RKP `hvc` invocation's first argument (X0) is the function identifier and as shown in (7) must abide by a specific format for function `rkp_main()` (0xb010d000) which is the `hvc` handler to be invoked. Specifically, each command is expected to have a prefix value of 0x83800000. Furthermore, to form the command, command indices are shifted by 12 and then ORed with the prefix (readers may also refer to kernel/include/linux/rkp.h). This format is also expected by `rkp_main()` as explained next.

```
// vmm-G955FXXU4CRJ5.elf

void rkp_main(unsigned int64_t command, exception_frame *exception_frame)
{

    hvc_cmd = (command >> 12) & 0xFF;                              /* (8) */

    if ( hvc_cmd && !is_rkp_activated )                            /* (9) */
```

```
        lead_to_policy_violation(hvc_cmd);
    ...

    my_check_hvc_command(hvc_cmd);
    switch ( hvc_cmd )
    {
        case 0:
            ...

            if ( is_rkp_activated )                              /* (10) */
                rkp_policy_violation(2, 0, 0, 0);

            rkp_init(exception_frame);
            ...

            break;
...


void my_check_hvc_command(unsigned int64_t cmd_index)
{
    if ( cmd_index > 0x9F )
        rkp_policy_violation(3, cmd_index, 0, 0);

    prev_counter = my_cmd_counter[cmd_index];

    if ( prev_counter != 0xFF )
    {
        cur_counter = (prev_counter - 1);

        if ( cur_counter > 1 )
            rkp_policy_violation(3, cmd_index, prev_counter, 0);

        my_cmd_counter[cmd_index] = cur_counter;
    }
}
```

`rkp_main()` first extracts the command index (8) and then calls function `my_check_hvc_command()` (0xb0113510). Two things are happening there. First, the index must be smaller than 0x9f. Second, RKP maintains an array with command counters. The counter for RKP initialization command is 1 during the array definition and is set again along with all other values at runtime in function `my_initialize_hvc_cmd_counter()` (0xb011342c) during the initialization. If any of these checks fails, `rkp_policy_violation()` (0xb010dba4) is called which can be considered as an assertion error and leads to system hang. Finally, before allowing any command invocation except for the initialization, a global flag indicating whether RKP is initialized is checked (9). This flag is obviously set after a successful initialization as explained in the following section.

Before continuing with the initialization process we will present some commands as examples to better demonstrate their usage. The first initialization function (presented next) is `rkp_init()` with command id 0 which corresponds to command 0x83800000. During definition, as mentioned above, its counter is set to 1 so that it can be called once before invoking `my_initialize_hvc_cmd_counter()`. Similarly, command id 1 corresponds to deferred initialization function (also presented next), can be reached with command 0x83801000 and since its counter is set to 1 which means it can only be called once. Commands with counter value -1 as the ones shown in the table below for handling page tables (commands 0x21 and 0x22 for level 1 and 2 correspondingly) can be called arbitrary number of times.

Function	ID	Command	Counter
rkp_init	0x0	0x83800000	0
rkp_def_init	0x1	0x83801000	1

```
...
| rkp_pgd_set  | 0x21 | 0x83821000 |      -1 |
| rkp_pmd_set  | 0x22 | 0x83822000 |      -1 |
...
```

------[2.3.2 - RKP Initialization

With this information, we are now ready to initialize RKP. In the snippet below we demonstrate the framework process to initialize the RKP (with RKP command id 0). We also show the `rkp_init_t` struct values used in the framework during the invocation and we will be elaborating more on them while examining the RKP initialization function `rkp_init()` (0xb0112f40). Interested readers can also study and compare `framework_rkp_init()` function with Samsung kernel function `rkp_init()` in kernel/init/main.c and the initialization values presented here against some of the values from the sample sec_log output above.

```
// framework/main.c

void el1_main(void) {
    framework_rkp_init();
    ...
}

// framework/vmm.h

#define RKP_PREFIX  (0x83800000)
#define RKP_CMDID(CMD_ID)  (((CMD_ID) << 12 ) | RKP_PREFIX)

#define RKP_INIT                RKP_CMDID(0x0)
...

// framework/vmm.c

void framework_rkp_init(void)
{
    struct rkp_init_t init;
    init.magic = RKP_INIT_MAGIC;
    init._text = (uint64_t)__va(&_text);
    init._etext = (uint64_t)__va(&_etext);
    init.rkp_pgt_bitmap = (uint64_t)&rkp_pgt_bitmap;
    init.rkp_dbl_bitmap = (uint64_t)&rkp_map_bitmap;
    init.rkp_bitmap_size = 0x20000;

    init.vmalloc_start = (uint64_t)__va(&_text);
    init.vmalloc_end = (uint64_t)__va(&_etext+0x1000);
    init.init_mm_pgd = (uint64_t)&swapper_pg_dir;
    init.id_map_pgd = (uint64_t)&id_pg_dir;
    init.zero_pg_addr = (uint64_t)&zero_page;
    init.extra_memory_addr = RKP_EXTRA_MEM_START;
    init.extra_memory_size = RKP_EXTRA_MEM_SIZE;
    init._srodata = (uint64_t)__va(&_srodata);
    init._erodata = (uint64_t)__va(&_erodata);

    rkp_call(RKP_INIT, &init, (uint64_t)VA_OFFSET, 0, 0, 0);
}

// framework/util.S

rkp_call:
    hvc #0
    ret
ENDPROC(rkp_call)
```

```
magic           : 0x000000005afe0001
vmalloc_start   : 0xffffff8080000000
```

```
vmalloc_end     : 0xffffff8080086000
init_mm_pgd     : 0x0000000080088000
id_map_pgd      : 0x000000008008b000
zero_pg_addr    : 0x000000008008e000
rkp_pgt_bitmap  : 0x0000000080044000
rkp_dbl_bitmap  : 0x0000000080064000
rkp_bitmap_size : 0x0000000000020000
_text           : 0xffffff8080000000
_etext          : 0xffffff8080085000
extra_mem_addr  : 0x00000000af400000
extra_mem_size  : 0x0000000000600000
physmap_addr    : 0x0000000000000000
_srodata        : 0xffffff8080085000
_erodata        : 0xffffff8080086000
large_memory    : 0x0000000000000000
fimc_phys_addr  : 0x00000008fa080000
fimc_size       : 0x0000000000780000
tramp_pgd       : 0x0000000000000000
```

Before everything else, the debug log at 0xb0200000 is initialized (11). This is the second EL2 oracle and we will be discussing it shortly as it will provide valuable information to help create correct memory mapping for the initialization to be successful.

Evidently, there are two modes of RKP operation which are decided upon during the initialization; normal and test mode. Test mode disables some of the aforementioned `hvc` command invocation counters and enables some command indices/functions. As the name suggests these are used for testing purposes and while these may assist and ease the reversing process, we will not be analyzing them in depth, because the are not encountered in real world setups. The mode is selected by the struct magic field, whose value can either be 0x5afe0001 (normal mode) or 0x5afe0002 (test mode).

It would be possible to change to test mode via a second `rkp_init()` invocation while hoping not to break any other configurations, however this is not possible via normal system interaction. As shown in (12) after a successful initialization, global flag `is_rkp_activated` is set. This flag is then checked (10) before calling `rkp_init()` in `rkp_main()` function as demonstrated in the previously presented snippet.

```
// vmm-G955FXXU4CRJ5.elf

void rkp_init(exception_frame *exception_frame)
{
    ...

    rkp_init_values = maybe_rkp_get_pa(exception_frame->x1);

    rkp_debug_log_init();                                        /* (11) */
    ...

    if ( rkp_init_values->magic - 0x5AFE0001 <= 1 ){

        if ( rkp_init_values->magic == 0x5AFE0002 )
        {
            /* enable test mode */
        }

        /* store all rkp_init_t struct values */

        rkp_physmap_init();

        ...

        if ( rkp_bitmap_init() )
        {
            /* misc initializations and debug logs */
            rkp_debug_log("RKP_6398d0cb", hcr_el2,
                             sctlr_el2, rkp_init_values->magic);

            /* more debug logs */

            if ( rkp_paging_init() )
            {
                is_rkp_activated = 1;                            /* (12) */
                ...

                my_initialize_hvc_cmd_counter();
                ...
            }
        }
        ...
    }
    ...
}
```

RKP maintains a struct storing all required information. During initialization in RKP function `rkp_init()`, values passed via `rkp_init_t` struct along with the VA_OFFSET are stored there to be used later. Next, various memory regions such as physmap and bitmaps are initialized. We are not going to be expanding on those regions since they are implementation specific, but due to their heavy usage by RKP (especially physmap) we are going to briefly explain them. Physmap contains information about physical regions, such as whether this is an EL2 or EL1 region etc., is set to a predefined EL2 only accessible region as explained next and RKP uses this information to decide if certain actions are allowed on specific regions.

Two bitmaps exist in this specific RKP implementation; rkp_pgt_bitmap and rkp_dbl_bitmap and their physical regions are provided by EL1 kernel. They are both written to by RKP. rkp_pgt_bitmap provides information to EL1 on whether addresses are protected by S2 mappings and as such accesses should be handled by RKP. rkp_dbl_bitmap is used to track and prevent unauthorized mappings from being used for page tables. The `rkp_bitmap_init()` success requires only the pointers to not be zero, however additional restrictions are defined during `rkp_paging_init()` function (0xb010e4c4) later.

Next, we see the RKP debug log being used, dumping system registers thus providing important information regarding the system state/configuration, which has helped us understand the system and configure the framework. Below a (processed) sample output is displayed with the various registers annotated. Finally, Samsung allows OEM unlock for the under examination device model, which allows us to patch vmm.elf, build and boot the kernel with the patched RKP and retrieve additional information. The final snippet line contains the debug log from a separate execution, where MAIR_ELn registers were replaced with SCTLR_EL1 and VTCR_EL2 respectively. How to build a custom kernel and boot a Samsung device with it is left as exercise to the reader.

```
0000000000000000    neoswbuilder-DeskTop RKP64_01aa4702
0000000000000000    Jul 11 2018
0000000000000000    11:19:42

/* hcr_el2 */      /* sctlr_el2 */
  84000003          30cd1835           5afe0001  RKP_6398d0cb

/* tcr_el2 */      /* tcr_el1 */
  80823518          32b5593519         5afe0001  RKP_64996474

/* mair_el2 */     /* mair_el1 */
21432b2f914000ff  0000bbff440c0400     5afe0001  RKP_bd1f621f
...
```

```
/* sctlr_el1 */    /* vtcr_el2 */
   34d5591d           80023f58         5afe0001  RKP_patched
```

Finally, one of the most important functions in RKP initialization follows; `rkp_paging_init()`. Numerous checks are performed in this function and the system memory layout must satisfy them all for RKP to by initialized successfully. Furthermore, physmap, bitmaps and EL2 Stage 1 and 2 tables are set or processed. We will be explaining some key points but will not go over every trivial check. Finally, we must ensure that any RKP required regions are reserved. The physical memory layout used in the framework aiming to satisfy the minimum requirements to achieve proper RKP initialization is shown below. Obviously, more complex layouts can be used to implement more feature rich frameworks.

The graph also explains the previously presented size selection of 3GBs for the emulation system RAM. This size ensures that the framework has a sufficiently large PA space to position executables in their expected PAs.

```
  +---------+ 0x80000000  text, vmalloc
  |         |
  |         |
  |         |
  |         |
  +---------+ 0x80044000  rkp_pgt_bitmap
  |         |
  |         |
  +---------+ 0x80064000  rkp_map_bitmap
  |         |
  |         |
  +---------+ 0x80085000  _etext, srodata
  |         |
  +---------+ 0x80086000  _erodata, vmalloc_end
  |         |
  |         |
  +---------+ 0x80088000  swapper_pg_dir
  |         |
  |         |
  +---------+ 0x8008b000  id_pg_dir
  |         |
  |         |
  +---------+ 0x8008e000  zero_page
  |         |
      ...
  |         |
  +---------+ 0xaf400000 rkp_extra_mem_start
  |         |
  |         |
  +---------+ 0xafa00000 rkp_extra_mem_end
  |         |
  +---------+ 0xafc00000 rkp_phys_map_start
  |         |
  |         |
  +---------+ 0xb0100000 rkp_phys_map_end, hyp_base
```

To sum up the process, after alignment and layout checks, the EL1 kernel region is set in physmap (13) and mapped in EL2 Stage 1 translation tables (14). The two bitmap regions are checked (15) and if they are not incorporated in the kernel text, their Stage 2 (S2) entries are changed to Read-Only and not executable (16) and finally physmap is updated with the two bitmap regions. FIMC region, which will be discussed shortly, is processed next (17) in function `my_process_fimc_region()` (0xb0112df0). Continuing, kernel text is set as RWX in S2 translation tables (18) which will change later during the initialization to read-only. Last but not least, physmap and extra memory address are unmapped from S2 (19) and (21) rendering them inaccessible from EL1 and their physmap regions are set (20) and (22).

```
// vmm-G955FXXU4CRJ5.elf

int64_t rkp_paging_init(void)
{
    /* alignment checks */


    v2 = my_rkp_physmap_set_region(text_pa, etext - text, 4);    /* (13) */
    if ( !v2 ) return v2;

    /* alignment checks */


    res = s1_map(text_pa, etext_pa - text_pa, 9);                /* (14) */
    ...

    /*
     * bitmap alignment checks                                   /* (15) */
     * might lead to label do_not_process_bitmap_regions
     */


    res = rkp_s2_change_range_permission(rkp_pgt_bitmap,         /* (16) */
            bitmap_size + rkp_pgt_bitmap, 0x80, 0, 1); // RO, XN
    ...

    res = rkp_s2_change_range_permission(rkp_map_bitmap,
                                            bitmap_size + rkp_map_bitmap,
                                            0x80, 0, 1); // RO, XN
    ...

do_not_process_bitmap_regions:

    if ( !my_rkp_physmap_set_region(rkp_pgt_bitmap, bitmap_size, 4) )
        return 0;

    res = my_rkp_physmap_set_region(rkp_map_bitmap, bitmap_size, 4);
    if ( res )
    {
        res = my_process_fimc_region();                          /* (17) */
        if ( res )
        {
            res = rkp_s2_change_range_permission(                /* (18) */
                                            text_pa, etext_pa,
                                            0, 1, 1);  // RW, X
            ...

                                                                 /* (19) */
            res = maybe_s2_unmap(physmap_addr, physmap_size + 0x100000);
            ...

            res = my_rkp_physmap_set_region(physmap_addr,        /* (20) */
                                            physmap_size + 0x100000, 8);
            ...

                                                                 /* (21) */
            res = maybe_s2_unmap(extra_memory_addr, extra_memory_size);
            ...

            res = my_rkp_physmap_set_region(extra_memory_addr,   /* (22) */
                                                extra_memory_size, 8);
            ...
        }
    }
    return res;
}
```

FIMC refers to Samsung SoC Camera Subsystem and during the kernel initialization, regions are allocated and binaries are loaded from the disk. There is only one relevant `hvc` call, related to the FIMC binaries verification (command id 0x71). RKP modifies the related memory regions permissions and then invokes EL3 to handle the verification in function `sub_B0101BFC()`. Since we are implementing our own EL3 and are interested in EL2 functionality we will be ignoring this region. However, we still reserve it for completeness reasons and function `my_process_fimc_region()` simply processes the S2 mappings for this region. By invoking `hvc` with command id 0x71, even if every other condition is met and `smc` is reached, as discussed above EL3 will hang because there is no handler for `smc` command id 0xc200101d in our setup.

```
// vmm-G955FXXU4CRJ5.elf

sub_B0101BFC

...
mov  X0, #0xC200101D
mov  X1, #0xC
mov  X2, X19  // holds info about fimc address, size, etc.
mov  X3, #0
dsb  SY
smc  #0
...
```

Although, as mentioned, simply reserving the region will suffice for this specific combination of hypervisor and subsystem, it is indicative of the considerations needed when examining hypervisors, even if more complex actions are required by other hypervisors and/or subsystems. For example the verification might have been incorporated in the initialization procedure, in which case this could be handled by our framework EL3 component.

At this step we have performed the first step of RKP initialization successfully. After some tasks such as the `hvc` command counters initialization and the `is_rkp_activated` global flag setting `rkp_init()` returns. We can now invoke other `hvc` commands.

------[2.3.3 - RKP Deferred Initialization

The next step is the deferred initialization which is handled by function `rkp_def_init()` (0xb01131dc) and its main purpose is to set the kernel S2 translation permissions.

```
// vmm-G955FXXU4CRJ5.elf

void rkp_def_init(void)
{
    ...

    if ( srodata_pa >= etext_pa )
    {
        if (!rkp_s2_change_range_permission(text_pa, etext_pa, 0x80, 1, 1))
            // Failed to make Kernel range ROX
            rkp_debug_log("RKP_ab1e86d9", 0, 0, 0);
    }
    else
    {
        res = rkp_s2_change_range_permission(text_pa, srodata_pa,
                                              0x80, 1, 1)) // RO, X
        ...

        res = rkp_s2_change_range_permission(srodata_pa, etext_pa,
                                              0x80, 0, 1)) // RO, XN
        ...
    }

    rkp_l1pgt_process_table(swapper_pg_dir, 1, 1);
    RKP_DISALLOW_DEBUG = 1;
    rkp_debug_log("RKP_8bf62beb", 0, 0, 0);
}
```

As demonstrated below after `rkp_s2_change_range_permission()` invocation the kernel region is set to read only. Finally, in `rkp_l1pgt_process_table()` swapper_pg_dir (TTBR1_EL1) and its subtables are set to read-only and not-executable.

```
// EL1 text before rkp_s2_change_range_permission()
Third level: 0x80000000-0x80001000: S2AP=11, XN=0
...
// EL1 text after rkp_s2_change_range_permission()
Third level: 0x80000000-0x80001000: S2AP=1, XN=0
...

// swapper_pg_dir before rkp_l1pgt_process_table()
Third level: 0x80088000-0x80089000: S2AP=11, XN=0
Third level: 0x80089000-0x8008a000: S2AP=11, XN=0
...
// swapper_pg_dir after rkp_l1pgt_process_table()
Third level: 0x80088000-0x80089000: S2AP=1, XN=10
Third level: 0x80089000-0x8008a000: S2AP=1, XN=10
...
```

------[2.3.4 - Miscellaneous Initializations

In our approach, we have not followed the original kernel initialization to the letter. Specifically, we skip various routines initializing values regarding kernel structs such as credentials, etc., which are void of meaning in our minimal framework. Moreover, these are application specific and do not provide any valuable information required by the ARM architecture to properly define the EL2 state. However, we will be briefly presenting them here for completeness reasons, and as our system understanding improves and the framework supported functionality requirements increase (for example to improve fuzzing discussed next) they can be incorporated in the framework.

Command 0x40 is used to pass information about cred and task structs offsets and then command 0x42 for cred sizes during the credential initialization in kernel's `cred_init()` function. Next, in `mnt_init()` command 0x41 is used to inform EL2 about vfsmount struct offsets and then when rootfs is mounted in `init_mount_tree()` information regarding the vfsmount are sent via command 0x55. This command is also used later for the /system partition mount. These commands can only be called once (with the exception of command 0x55 whose counter is 2) and as mentioned above are used in the original kernel initialization process. Incorporating them to the framework requires understanding of their usage from both the kernel and the hypervisor perspective which will be left as an exercise to the reader who can start by studying the various `rkp_call()` kernel invocations.

----[2.4 - Final Notes

At this point we have performed most of the expected RKP initialization routines. We now have a fully functional minimal framework which can be easily edited to test and study the RKP hypervisor behavior.

More importantly we have introduced fundamental concepts for readers to implement their own setups and reach the current system state which allows us to interact with it and start investigating fuzzing implementations.

On a final note, some of the original kernel initialization routines were omitted since their action lack meaning in our framework. They were briefly introduced and interested readers can study the various `rkp_call()` kernel invocations and alter the framework state at will. Additionally, this allows the fuzzers to investigate various configuration scenarios not restricted by our own assumptions.

--[3 - Fuzzing

In this section we will be describing our approaches towards setting up fuzzing campaigns under the setup presented above. We will begin with a naive setup aiming to introduce system concepts we need to be aware and an initial interaction with QEMU source code and functionality. We will then be expanding on this knowledge, incorporating AFL in our setup for more intelligent fuzzing.

To verify the validity of the fuzzing setups presented here we evidently require a bug that would crash the system. For this purpose we will be relying on a hidden RKP command with id 0x9b. This command leads to function `sub_B0113AA8()` which, as shown in the snippet, adds our second argument (register X1) to value 0x4080000000 and uses the result as an address to store a QWORD. As you might be imagining, simply passing 0 as our second argument results in a data abort ;)

```
// vmm-G955FXXU4CRJ5.elf

int64_t sub_B0113AA8(exception_frame *exc_frame)
{
    *(exc_frame->x1 + 0x4080000000) = qword_B013E6B0;
    rkp_debug_log("RKP_5675678c", qword_B013E6B0, 0, 0);
    return 0;
}
```

To demonstrate the framework usage we are going to trigger this exception with a debugger attached. We start the framework and set a breakpoint in the handler from `hvc` command 0x9b at the instruction writing the QWORD to the evaluated address. Single stepping from there causes an exception, which combined with the previous information about RKP exception handlers, we can see is a synchronous exception from the same EL. Continuing execution from there we end up in the synchronous handler for data aborts (EC 0x25) which leads to `vmm_panic()` :)

```
(gdb) target remote :1234
_reset () at boot64.S:15
15          ldr x30, =stack_top_el3

(gdb) continue
...
Breakpoint 1, 0x00000000b0113ac4 in ?? ()
(gdb) x/4i $pc-0x8
   0xb0113abc:  mov     x0, #0x80000000
   0xb0113ac0:  movk    x0, #0x40, lsl #32
=> 0xb0113ac4:  str     x1, [x2,x0]
   0xb0113ac8:  adrp    x0, 0xb0116000
(gdb) info registers x0 x1 x2
x0             0x4080000000     277025390592
x1             0x0      0
x2             0x1      1


(gdb) stepi
0x00000000b010c1f4 in ?? ()
```

```
(gdb) x/20i $pc
=> 0xb010c1f4:  stp     x1, x0, [sp,#-16]!
   ...
   0xb010c234:  mov     x0, #0x200    // Current EL
   0xb010c238:  mov     x1, #0x0      // Synchronous
   0xb010c23c:  mov     x2, sp
   0xb010c240:  bl      0xb010aa44    // vmm_dispatch

(gdb) continue
Continuing.


Breakpoint 5, 0x00000000b010a80c in ?? ()  // EC 0x25 handler
(gdb) x/7i $pc
=> 0xb010a80c:  mov     x0, x22
   0xb010a810:  mov     x1, x21
   0xb010a814:  mov     x2, x19
   0xb010a818:  adrp    x3, 0xb0115000
   0xb010a81c:  add     x3, x3, #0x4d0
   0xb010a820:  bl      0xb010a4cc    // vmm_panic
```

----[3.1 - Dummy fuzzer

To implement the dummy fuzzer we decided to abuse `brk` instruction, which generates a Breakpoint Instruction exception. The exception is recorded in in ESR_ELx and the value of the immediate argument in the instruction specific syndrome field (ESR_ELx.ISS, bits[24:0]). In QEMU, this information is stored in `CPUARMStame.exception` structure as shown in the following snippet.

```
// qemu/target/arm/cpu.h

typedef struct CPUARMState {
    ...

    /* Regs for A64 mode.  */
    uint64_t xregs[32];

    ...

    /* Information associated with an exception about to be taken:
     * code which raises an exception must set cs->exception_index and
     * the relevant parts of this structure; the cpu_do_interrupt function
     * will then set the guest-visible registers as part of the exception
     * entry process.
     */
    struct {
        uint32_t syndrome; /* AArch64 format syndrome register */
        ...
    } exception;
    ...
}
```

`arm_cpu_do_interrupt()` function handles the exceptions in QEMU and we can intercept the `brk` invocation by checking `CPUState.exception_index` variable as shown in (23). There we can introduce our fuzzing logic and setup the system state with our fuzzed values for the guest to access as discussed next. Finally, to avoid actually handling the exception (calling the exception vector handle, changing ELs etc.) which would disrupt our program flow, we simply advance `pc` to the next instruction and return from the function. This effectively turns `brk` into a fuzzing instruction.

```
// qemu/target/arm/helper.c

/* Handle a CPU exception for A and R profile CPUs.
```

```
    ...
 */
void arm_cpu_do_interrupt(CPUState *cs)
{
    ARMCPU *cpu = ARM_CPU(cs);
    CPUARMState *env = &cpu->env;
    ...

    // Handle the break instruction
    if (cs->exception_index == EXCP_BKPT) {                      /* (23) */

        handle_brk(cs, env);

        env->pc += 4;
        return;
    }
    ...

    arm_cpu_do_interrupt_aarch64(cs);
    ...
}
```

We utilize syndrome field as a function identifier and specifically immediate value 0x1 is used to call the dummy fuzzing functionality. There are numerous different harnesses that can be implemented here. In our demo approach we only use a single argument (via X0) which points to a guest buffer where fuzzed data could be placed. The framework registers, hence arguments which will be passed to EL2 by `rkp_call_fuzz` after calling `__break_fuzz()` are set by our harness in function `handle_brk()`.

```
// framework/main.c

void el1_main(void) {
    framework_rkp_init();
    rkp_call(RKP_DEF_INIT, 0, 0, 0, 0, 0);

    for(; ;){ // fuzzing loop
        __break_fuzz(); // create fuzzed values
        rkp_call_fuzz(); // invoke RKP
    }
}

// framework/util.S

__break_fuzz:
    ldr x0, =rand_buf
    brk #1
    ret
ENDPROC(__break_fuzz)

rkp_call_fuzz:
    hvc #0
    ret
ENDPROC(rkp_call_fuzz)
```

We will not be presenting complex harnesses here since this is beyond the scope of this article and will be left as exercise for the reader. We will, however, be describing a simple harness to fuzz RKP commands. Moreover, since most RKP handlers expect the second argument (X1 register) to point to a valid buffer we will be using `rand_buf` pointer as shown above for that purpose.

The logic should be rather straightforward. We get a random byte (24), at the end place it in X0 (25) and as a result will be used as the RKP command index. Next, we read a page of random data and copy it to the guest buffer `rand_buf` (using function `cpu_memory_rw_debug()`) and use it as the second argument by placing the buffer address in X1 (26).

```
// qemu/target/arm/patch.c

int handle_brk(CPUState *cs, CPUARMState *env)
{
    uint8_t syndrome = env->exception.syndrome & 0xFF;
    int l = 0x1000;
    uint8_t buf[l];

    switch (syndrome) {
        case 0: // break to gdb
            if (gdbserver_running()) {
                qemu_log_mask(CPU_LOG_INT, "[!] breaking to gdb\n");
                vm_stop(RUN_STATE_DEBUG);
            }
            break;
        case 1: ; // dummy fuzz
            uint8_t cmd = random() & 0xFF;                       /* (24) */

            /* write random data to buffer buf */

            /*
             * Write host buffer buf to guest buffer pointed to
             * by register X0 during brk invocation
             */
            if (cpu_memory_rw_debug(cs, env->xregs[0], buf, l, 1) < 0) {
                fprintf(stderr, " Cannot access memory\n");
                return -1;
            }

            fuzz_cpu_state.xregs[0] = 0x83800000 | (cmd << 12);
            fuzz_cpu_state.xregs[1] = env->xregs[0];


            env->xregs[0] = fuzz_cpu_state.xregs[0];             /* (25) */
            env->xregs[1] = fuzz_cpu_state.xregs[1];             /* (26) */
            break;
        default:
            ;
    }
    return 0;
}
```

As you might expect after compiling the modified QEMU and executing the fuzzer, nothing happens! We elaborate more on this next.

------[3.1.1 - Handling Aborts

Since this is a bare metal implementation there is nothing to "crash". Once an abort happens, the abort exception handler is invoked and both our framework and RKP ends up in an infinite loop. To identify aborts we simply intercept them in `arm_cpu_do_interrupt()` similarly with `brk`.

```
// qemu/target/arm/helper.c

void arm_cpu_do_interrupt(CPUState *cs)
{
    ...

    // Handle the instruction or data abort
    if (cs->exception_index == EXCP_PREFETCH_ABORT ||
          cs->exception_index == EXCP_DATA_ABORT ) {

        if(handle_abort(cs, env) == -1) {
            qemu_system_shutdown_request(SHUTDOWN_CAUSE_HOST_ERROR);
        }
        // reset system
        qemu_system_reset_request(SHUTDOWN_CAUSE_HOST_QMP_SYSTEM_RESET);
    }
    ...
}
```

When a data or instruction abort exception is generated, we create a crash log in `handle_abort()` and then request QEMU to either reset and restart fuzzing or terminate if `handle_abort()` fails which essentially terminates fuzzing as we can not handle aborts. We use QEMU functions to dump the system state such as the faulting addresses, system registers, and memory dumps in text log files located in directory crashes/.

```
int handle_abort(CPUState *cs, CPUARMState *env)
{
    FILE* dump_file;

    if (open_crash_log(&dump_file) == -1) return -1;


    const char *fmt_str = "********* Data\\Instruction abort! *********\n"
                          "FAR = 0x%llx\t ELR = 0x%llx\n"
                          "Fuzz x0 = 0x%llx\t Fuzz x1 = 0x%llx\n";

    fprintf(dump_file, fmt_str, env->exception.vaddress,
                                env->pc,
                                fuzz_cpu_state.xregs[0],
                                fuzz_cpu_state.xregs[1]);

    fprintf(dump_file, "\n********** CPU State **********\n");
    cpu_dump_state(cs, dump_file, CPU_DUMP_CODE);
    fprintf(dump_file, "\n********** Disassembly **********\n");
    target_disas(dump_file, cs, env->pc-0x20, 0x40);
    fprintf(dump_file, "\n********** Memory Dump **********\n");
    dump_extra_reg_data(cs, env, dump_file);

    fprintf(dump_file, "\n********** End of report **********\n");

    fclose(dump_file);

    return 0;
}
```

A sample trimmed crash log is presented below. We can see that the faulting command is 0x8389b000 (or command index 0x9b ;) the faulting address and the code were the abort happened. You can create your own logs by executing the dummy fuzzer ;)

```
********** Data\Instruction abort! **********
FAR = 0x41000c5000       ELR = 0xb0113ac4
Fuzz x0 = 0x8389b000     Fuzz x1 = 0x800c5000

********** CPU State **********
 PC=00000000b0113ac4 X00=0000004080000000 X01=0000000000000000
X02=00000000800c5000 X03=0000000000000000 X04=0000000000000000
....
X29=00000000b0142e70 X30=00000000b010d294  SP=00000000b0142e70
PSTATE=600003c9 -ZC- NS EL2h

********** Disassembly **********
0xb0113abc:  d2b00000  movz     x0, #0x8000, lsl #16
0xb0113ac0:  f2c00800  movk     x0, #0x40, lsl #32
0xb0113ac4:  f8206841  str      x1, [x2, x0]
0xb0113ac8:  f0000000  adrp     x0, #0xb0116000
```

```
0xb0113acc:  911ac000  add      x0, x0, #0x6b0

********** Memory Dump **********
...
X00: 0x0000004080000000
000000407fffff60: Cannot access memory

...

X02: 0x00000000800c5000
...
00000000800c4fe0: 0x0000000000000000 0x0000000000000000
00000000800c4ff0: 0x0000000000000000 0x0000000000000000
00000000800c5000: 0x21969a71a5b30938 0xc6d843c68f2f38be
00000000800c5010: 0xd7a1a2d7948ffd7e 0x42793a9f98647619
00000000800c5020: 0x87c01b08bb98d031 0x1949658c38220d4d

...

********** End of report **********
```

------[3.1.2 - Handling Hangs

RKP has two functions that lead to system hangs; `rkp_policy_violation()` and `vmm_panic()`. The former is used when RKP unsupported exceptions or exception classes are triggered, while the latter aligns better with the `assert()` function logic.

Since there are only two functions with these characteristics we can simply reset the system if they are ever executed. This is done in QEMU function `cpu_tb_exec()` which is responsible for emulating the execution of a single basic block. When they are identified via their address, the system is reset as with the abort case presented above, without however creating a crash log file.

Evidently, this is not an optimal approach and does not scale well. We will be providing a better solution in the setup with AFL described next.

```
// qemu/accel/tcg/cpu-exec.c

/* Execute a TB, and fix up the CPU state afterwards if necessary */
static inline tcg_target_ulong cpu_tb_exec(CPUState *cpu,
                                           TranslationBlock *itb)
{
    CPUArchState *env = cpu->env_ptr;
    ...

    if (env->pc == 0xB010DBA4) { // rkp_policy_violation
        qemu_log("[!] POLICY VIOLATION!!! System Reset!\n");
        qemu_system_reset_request(SHUTDOWN_CAUSE_HOST_QMP_SYSTEM_RESET);
    }

    if (env->pc == 0xB010A4CC) { // vmm_panic
        qemu_log("[!] VMM PANIC!!! We should not be here!!!\n");
        qemu_system_reset_request(SHUTDOWN_CAUSE_HOST_QMP_SYSTEM_RESET);
    }

    ...
}
```

----[3.2 - AFL with QEMU full system emulation

One of the major problems encountered during this work was QEMU changing rapidly. This caused various tools to become obsolete, unless teams were dedicated porting them to newer versions fixing various problems introduced by the modified QEMU code. With this in mind, we will first introduce problems stemming from this situation and previous work on full system emulation. We will then proceed with the proposed solution.

------[3.2.1 - Introduction

As mentioned before, we chose the latest stable QEMU v4.1.0 and AFL v2.56b. The first step was to port AFL to the target QEMU version. The patch itself is rather straightforward, so we will not be presenting details here. Refer to the attached afl-2.56-qemu-4.1.0-port/readme for more details. Note that to remove the QEMU directory from the AFL subfolder, we included in AFL header files config.h and afl-types.h in the patch. As a result, to avoid any unexpected behaviors theses files must be kept in sync between AFL and QEMU.

After applying the patches and building QEMU and copying the resulting binary in AFL directory as `afl-qemu-trace`, we invoke AFL with QEMU in the old fashioned way:

```
$ ./afl-fuzz -Q -i in -o out /usr/bin/readelf -a @@
```

We will briefly explain some QEMU/AFL key points to help understand the modified version. With QEMU the forkserver practically runs inside QEMU, starts when the ELF entry point is encountered and is kept in sync with AFL via pipes. When AFL instructs forkserver to run once, the forkserver (parent) clones itself, writes the QEMU child (child) pid to AFL and allows the child to execute free. AFL sets a child execution watchdog which will terminate the child if triggered. While the child runs it updates the AFL bitmap (`afl_maybe_log()`) and reports blocks that have not been translated yet back to the parent (`afl_request_tsl()`) who waits in a read loop (`afl_wait_tsl()`). Once a new block is encountered the parent mirrors the translation and avoid re-translation for future children which significantly improves fuzzing performance (interested readers can also check [07]). Upon termination/crash/exit of the child, parent exits the wait loop, reports back to AFL and awaits AFL to order a new execution.

```
   +-------+              +-------------+            +------------+
   |  AFL  |              | Qemu Parent |            | Qemu Child |
   +-------+              +-------------+            +------------+
       |                          .                        .
 init_forkserver                  .                        .
       |                          .                        .
   fork/exec  ------------>  afl_setup                     .
       |                   (entry point)                   .
   setitimer                      |                        .
       |                          |                        .
     read    <----+               |                        .
    (block)       |        afl_forkserver                  .
       |          |               |                        .
       |          +--unblock--- write                      .
       |                          |    <--------------------------------+
  run_target          +-------> read                       .           |
       |              |        (block)                     .           |
       |              |           |                        .           |
     write --unblock--+           |                        .           |
       |                          |                        .           |
     read    <----+             fork  ----------------->  run          |
    (block)       |               |                        | <------+  |
       |          |               |                        |        |  |
       |          +--unblock--- write                afl_maybe_log  |  |
   setitimer                 (child pid)                   |        |  |
       |                          |                        |        |  |
     read   <-----+               |                        |        |  |
    (block)       |               |                        |        |  |
       |          |       afl_wait_tsl/read  <----- afl_request_tsl |  |
       |          |         (loop block)             write          |  |
       |          |               |                        |        |  |
    do stuff      |               |                        +--------+  |
       |          |           waitpid() <---+              |           |
       |          |               |         |          terminate       |
       |          |               |         +-----------  exit         |
       |          |               |                      crash         |
       |          +--unblock--- write                                  |
       |                 (child status)                                |
       |                          |                                    |
       |                          +--------------repeat----------------+
```

Our approach is based on TriforceAFL [08] whose goal was to fuzz the Linux kernel. We are going to provide a brief overview but skip various details, because as aforementioned TriforceAFL is based on old QEMU (2.3.0) and AFL (2.06b) versions, currently does not build and the project seems to be abandoned. Furthermore, Linux kernel is vastly more complex compared to our framework and the targeted hypervisor and for this reason different hashing algorithm for the bitmap was used, which is not required here. Additionally, the target in this article is an ARM binary and executes on different level (EL2) from the Linux kernel (EL1). Nonetheless, interested readers may refer to the project source code, documentation [09] and slides for additional details.

In short, they introduced an instruction as a handler to dispatch operations to 4 different functions called "hypercalls", all handled by QEMU. The parent executes normally and boots the VM until the first hypercall `startForkServer` is encountered which causes the forkserver to be instantiated. The parent/forkserver the spawns a child guest which then invokes hypercall `getWork` to fetch the new testcase from the host to the guest VM and then hypercall `startWork` to enable tracing and set the address region to be traced. If the child does not crash, it terminates by calling hypercall `endWork` to force the child QEMU to exit. These "hypercalls" are invoked from a custom Linux kernel driver.

As stated in TriforceAFL, getting forkserver to work was one of the most difficult parts. QEMU full system emulation uses 3 threads; CPU, IO and RCU. Their solution was to have `startForkServer` hypercall set a flag which causes CPU thread (vCPU) to exit the CPU loop, save some state information, notify the IO thread and exit. IO thread then receives the notification and starts the forkserver by forking itself. The child IO thread then spawns a new vCPU thread which restores its state from the previous vCPU saved information and continues execution cleanly from `startForkServer`. Basically, the forkserver is the IO thread (whose vCPU thread has now terminated) and each new fork child spawns a new vCPU thread (with information from the parent vCPU saved state) to do the CPU emulation.

Finally, AFL was edited to increase the QEMU parent/child memory limit MEM_LIMIT_QEMU because full system emulation has larger memory requirements compared to user mode emulation, especially for emulating Linux kernel. Furthermore, during the AFL `init_forkserver()` fork, a timer controlled by FORK_WAIT_MULT defined value is set in AFL to avoid blocking in read indefinitely in case forkserver in parent fails. This value was increased, because during this step the parent initializes the guest VM until `startForkServer` hypercall is reached, which might be time consuming. Last but not least, mode enabled by argument -QQ was introduced to allow user to specify the QEMU binary path instead of using `afl-qemu-trace`.

Our approach relies heavily on TriforceAFL as mentioned before. We decided to skip the TriforceAFL implementation details due to the vast QEMU differences, however we recommend readers to study the TriforceAFL [08] implementation and documentation.

------[3.2.2 - Implementation

First we are going to go over the AFL diff which is the most brief since we only modified afl-fuzz.c and config.h and we do not deviate much from TriforceAFL. The QEMU parent/child memory limits have been commented out since our framework emulation has much larger memory requirements in comparison. Secondly, to disable QEMU chaining feature which affects AFL stability, AFL normally sets environmental variable "QEMU_LOG" to "nochain" (see qemu/linux-user/main.c for details) before invoking QEMU in user mode. This option however is not honored in full system emulation and as a result QEMU option `-d nochain` _must_ be specified during QEMU full system emulation invocation. Lastly, users must set the various system configurations AFL requires such as disabling the CPU frequency scaling and external core dump handling utilities. We invoke the fuzzer with our setup as:

```
$ AFL_SKIP_CPUFREQ=1 AFL_I_DONT_CARE_ABOUT_MISSING_CRASHES=1 \
    ./afl-fuzz -QQ -i in -o out                              \
      <path-to-qemu>/aarch64-softmmu/qemu-system-aarch64     \
        -machine virt                                        \
        -cpu cortex-a57                                      \
        -smp 1                                               \
        -m 3G                                                \
        -kernel kernel.elf                                   \
        -machine gic-version=3                               \
        -machine secure=true                                 \
        -machine virtualization=true                         \
        -nographic                                           \
        -d nochain                                           \
        -afl_file @@
```

--------[3.3.2.1 - QEMU patches

At this point we will be providing details regarding the QEMU patches to support full system AFL fuzzing since as mentioned before, even though the main idea persists, there are many differences compared to the original TriforceAFL implementation mainly due to vast QEMU differences between the versions. The first difference is that we utilized `brk` to introduce hypercalls instead of introducing a new instruction.

```
// qemu/target/arm/patch.c

int handle_brk(CPUState *cs, CPUARMState *env)
{
    ...

    switch (syndrome) {
        ...

        case 3:
            return start_forkserver(cs, env, env->xregs[0]);
        case 4:
            return get_work(cs, env, env->xregs[0], env->xregs[1]);
        case 5:
            return start_work(cs, env, env->xregs[0], env->xregs[1]);
        case 6:
            return done_work(env->xregs[0]);
        default:
            ;
    }
    return 0;
}
```

To better demonstrate the setup we provide the following diagram and each step will be explained next. Readers are also advised to compare this with the original AFL/QEMU diagram presented previously.

```
    +-------------+       +-------------+   +------------+  +-------------+
    | Qemu Parent |       | Qemu Parent |   | Qemu Child |  | Qemu  Child |
    |  IO thread  |       | vCPU thread |   |  IO thread |  | vCPU thread |
    +-------------+       +-------------+   +------------+  +-------------+
           |                     .                .               .
       initialize                .                .               .
          QEMU                   .                .               .
           |                     .                .               .
  (27) start vCPU -----> thread entry point       .               .
           |                     |                .               .
        do stuff <-+   tcg_register_thread (28)   .               .
           |       |             |                .               .
           +-------+             |                .               .
           |             main execution loop      .               .
           |              execute guest VM        .               .
           |            until start_forkserver    .               .
           |                    (29)              .               .
           |                     |                .               .
           |                     |                .               .
           |              start_forkserver        .               .
           |                     |                .               .
           |           set afl_wants_cpu_to_stop  .               .
           |                    (30)              .               .
           |                     |                .               .
           |              save vCPU state         .               .
           |                    (31)              .               .
           |                     |                .               .
           |               +-- notify             .               .
           |               | IO thread            .               .
           |               |    (32)              .               .
           |               |     |                .               .
  got_pipe_notification <--+   exit               .               .
           |                     |                .               .
    afl_forkserver (33)          X                .               .
           |                                      .               .
    write(unblock AFL)                            .               .
           |                                      .               .
 +-> read(from AFL, block)                        .               .
 |         |                                      .               .
 |       fork -------------------------> restore vCPU state      .
 |         |                                     (34)             .
 |         |                                      |               .
 |         |                                   start --> thread entry point
 |         |                                  vCPU (35)           |
 |         |                                      |               |
 |         |                                      |     tcg_register_thread
 |         |                                      |             (36)
 |         |                                      |               |
 |       write                                    |            getWork
 |  (child pid to AFL)                            |               |
 |         |                              +--> do stuff           |
 repeat   ...                             |       |           startWork
 |         |                              +-------+               |
 |         |                                                      |
 |    afl_wait_tsl  <-----------------+                      afl_maybe_log
 |        (37)                        |                           |
 |         |                          |                           |
 |         |                          +-------------------  afl_request_tsl
 |      waitpid <-----------+                                   (38)
 |         |                |                                     |
 |         |                |                                     |
 |       write              |                                   crash
 | (child status to AFL)    +--------------------------------  endWork
 |         |
 +---------+
```

During system initialization, vCPU is instantiated (27) by IO thread in a manner dependent on the system configuration. Our setup uses Multithread Tiny Coge Generator (MTTCG) which allows the host to run one host thread per guest vCPU. Note that we are using a single core/thread and as a result there is a single vCPU thread in our setup.

The vCPU thread entry point for MTTCG configuration is function `qemu_tcg_cpu_thread_fn()` under qemu/cpus.c where, after some initializations, vCPU enters its main execution loop (29)-(40). In a high level of abstraction, execution loop comprises two steps; translating basic blocks (function `tb_find()`) and executing them (function `cpu_tb_exec()`).

As mentioned before, we allow the QEMU parent to execute free and initialize the guest VM until `start_forkserver` hypercall is invoked. As a result, each forkserver child will start with a _fully initialized VM_ right before the targeted functionality significantly improving fuzzing performance.

```
// qemu/cpus.c

/* Multi-threaded TCG
 *
 * In the multi-threaded case each vCPU has its own thread. The TLS
 * variable current_cpu can be used deep in the code to find the
 * current CPUState for a given thread.
 */

static void *qemu_tcg_cpu_thread_fn(void *arg)
{
    CPUState *cpu = arg;

    ...

    tcg_register_thread();                                  /* (39) */

    do {
        ...

        r = tcg_cpu_exec(cpu);                              /* (40) */

        ...

    } while ((!cpu->unplug || cpu_can_run(cpu))             /* (41) */
               && !afl_wants_cpu_to_stop);

    if(afl_wants_cpu_to_stop) {
        ...

        if(write(afl_qemuloop_pipe[1], "FORK", 4) != 4)     /* (42) */
            perror("write afl_qemuloop_pip");
        ...

        restart_cpu = (&cpus)->tqh_first;                   /* (43) */

        ...
    }
    ...

    return NULL;
}
```

When during the execution `start_forkserver()` hypercall is invoked, global flag `afl_wants_cpu_to_stop` is set (30)-(44) ultimately breaking the vCPU main execution loop. There are various reasons that could cause the system to reach this state so after the main loop we check flag `afl_wants_cpu_to_stop` to decide whether vCPU must terminate (41). Finally we save the vCPU state (31)-(43), notify IO thread (32)-(42) and terminate the vCPU thread.

```
// qemu/target/arm/patch.c

target_ulong start_forkserver(CPUState* cs, CPUARMState *env, ...)
{
    ...

    /*
     * we're running in a cpu thread. we'll exit the cpu thread
     * and notify the iothread. The iothread will run the forkserver
     * and in the child will restart the cpu thread which will continue
     * execution.
     */
    afl_wants_cpu_to_stop = 1;                                    /* (44) */

    return 0;
}
```

Parent IO thread becomes the forkserver in the notification handling function `got_pipe_notification()` (33)-(45). In the fork child (which is the child QEMU IO thread) we reset the vCPU state (34)-(46) and start a new vCPU thread for the child process (35)-(47). (don't forget to comment out the `madvise(..., DONTFORK)` invocation ;)

```
// qemu/cpus.c

static void got_pipe_notification(void *ctx)
{
    ...

    afl_forkserver(restart_cpu);                                  /* (45) */


    /* we're now in the child! */
    (&cpus)->tqh_first = restart_cpu;                             /* (46) */

    qemu_tcg_init_vcpu(restart_cpu);                              /* (47) */
}
```

Finally, for MTTCG all TCG threads must register their context before starting translation (36)-(39) as part of their initialization process mentioned before. As shown next, each thread registers its context in `tcg_ctxs` array in an incremental fashion and assigns it to thread local variable `tcg_ctx`. It is obvious that the system was not designed with a forkserver in mind, where vCPU thread is respawned and trying to register a new context for the forkserver children will fail. However, since we use a single thread and we can simply bypass this by patching function `tcg_register_thread()` to always set `tcg_ctx` to the first array entry after the first invocation.

```
// qemu/tcg/translate-all.c

__thread TCGContext *tcg_ctx;


// qemu/tcg/tcg.c

void tcg_register_thread(void)
{
    static bool first = true;
    if (!first) {
        tcg_ctx = tcg_ctxs[0];
        return;
    }
    first = false;
    ...

    *s = tcg_init_ctx;

    ...

    /* Claim an entry in tcg_ctxs */
    n = atomic_fetch_inc(&n_tcg_ctxs);
    g_assert(n < ms->smp.max_cpus);
    atomic_set(&tcg_ctxs[n], s);

    tcg_ctx = s;

    ...
}
```

--------[3.3.2.2 - Framework support

Let's now demonstrate how to reach the state where forkserver is up and running via the framework. After the framework initialization we call `__break_start_forkserver()` from EL1 (48) which in turn calls `brk` with instruction specific syndrome 3 which corresponds to the `start_forkserver` hypercall. This eventually causes the forkserver to be started in the parent QEMU process as discussed above.

Each new child QEMU process, will resume guest VM execution in its vCPU at the instruction immediately following `__break_start_forkserver()` in a guest VM state identical to the one the parent process had before instantiating the forkserver. For example, in our setup the child will continue in (49) invoking the `get_work` hypercall to fetch the test case from the host (technically it will resume from `ret` instruction after `brk #3` in `__break_start_forkserver()` function but you get the idea ;).

```
// framework/main.c

void el1_main(void) {
    framework_rkp_init();

    rkp_call(RKP_DEF_INIT, 0, 0, 0, 0, 0);


    __break_start_forkserver(0);                                 /* (48) */

    /* fuzzing loop */
    for(; ;){
        __break_get_work();                                      /* (49) */
        __break_start_work();

        rkp_call_fuzz_afl((*(uint64_t*)(&rand_buf)), &rand_buf); /* (50) */

        __break_end_work(0);
    }
}


// framework/afl.S

__break_start_forkserver:
    brk #3
    ret
ENDPROC(__break_start_forkserver)

__break_get_work:
    ldr x0, =rand_buf
    mov x1, 0x1000
    brk #4
    ret
```

```
ENDPROC(__break_get_work)

__break_start_work:
    mov   x0, #RKP_VMM_START
    add   x1, x0, #RKP_VMM_SIZE
    brk #5
    ret
ENDPROC(__break_start_work)

rkp_call_fuzz_afl:
    hvc #0
    ret
ENDPROC(rkp_call_fuzz_afl)

__break_end_work:
    // x0 is the exit value
    brk #6
    ret
ENDPROC(__break_end_work)
```

For demonstration purposes and to verify that the fuzzer works as expected, we will be using the same fuzzing harness as with the dummy fuzzer to fuzz the `hvc` command ids. If everything works as expected we should have at least one crash by invoking command 0x9b.

As mentioned above, framework function `__break_get_work()` (49) invokes qemu `get_work` hypercall (51). There, the child QEMU reads the AFL created test case and copies its contents in guest VM `rand_buf`. In the next step, `__break_start_work()` framework function invokes `start_work` hypercall (52) which sets the child process to only track and edit the AFL bitmap for addresses in the RKP range.

```
// qemu/target/arm/patch.c

static target_ulong get_work(CPUState *cs, CPUARMState *env,     /* (51) */
                             target_ulong guest_ptr, target_ulong sz)
{
    int l = 0x1000;
    uint8_t buf[l];

    assert(afl_start == 0);
    fp = fopen(afl_file, "rb");
    if(!fp) {
         perror(afl_file);
         return -1;
    }

    fread(buf, l, 1, fp); // must add checks

    if (cpu_memory_rw_debug(cs, guest_ptr, buf, l, 1) < 0) {
        fprintf(stderr, " Cannot access memory\n");
        return -1;
    }


    fclose(fp);
    return retsz;
}


static target_ulong start_work(CPUState *cs, CPUArchState *env,  /* (52) */
                               target_ulong start, target_ulong end)
{
    afl_start_code = start;
    afl_end_code   = end;
    afl_start = 1;
    return 0;
}
```

The initial testcase provided to AFL must execute without crashing. For that we use command id 0x98 which as shown in the snippet simply writes in the debug log and exits. At long last, we can invoke and fuzz the `hvc` handler. We read the first QWORD (50) from the provided test case as the command id and simply use `rand_buf` as the second argument as discussed in the dummy fuzzer harness.

```
// vmm-G955FXXU4CRJ5.elf

void rkp_main(uint64_t command, exception_frame *exception_frame)
{
  ...

  switch ( hvc_cmd )
  {
    ...

    case 0x98:
      rkp_debug_log("RKP_a3d40901", 0, 0, 0); // CFP_JOPP_INIT
      break;
...
```

However, not long after the `hvc` invocation our system crashes. The problem lies in the basic block translations performed by the QEMU parent process as we elaborate on in the next section.

--------[3.3.2.3 - Handling parent translations

For QEMU to perform basic block translations for ARM architectures, it uses `mmu_idx` to distinguish translation regimes, such as Non-Secure EL1 Stage 1, Non-Secure EL1 Stage 2 etc. (for more details refer to ARMMMUIdx enum definition under qemu/target/arm/cpu.h). As shown below, to evaluate the current `mmu_idx` it relies on the current CPU PSTATE register (53). This process is normally performed by the vCPU thread during the guest VM emulation.

```
// qemu/target/arm/helper.c

int cpu_mmu_index(CPUARMState *env, bool ifetch)
{
    return arm_to_core_mmu_idx(arm_mmu_idx(env));
}

ARMMMUIdx arm_mmu_idx(CPUARMState *env)
{
    int el;
    ...

    el = arm_current_el(env);
    if (el < 2 && arm_is_secure_below_el3(env)) {
        return ARMMMUIdx_S1SE0 + el;
    } else {
        return ARMMMUIdx_S12NSE0 + el;
    }
}


// qemu/target/arm/cpu.h

static inline int arm_current_el(CPUARMState *env)
{
    ...
    if (is_a64(env)) {
        return extract32(env->pstate, 2, 2);                    /* (53) */
    }
    ...
}
```

As earlier discussed, in QEMU/AFL when a child process encounters a basic block previously not translated, it instructs (38)-(55) the parent to mirror the basic block translation process (37)-(57) so that next children will have cached copies to avoid re-translation and improve performance [07]. To achieve this, the child sends (55) the current pc address along with other information for the parent to perform the translation (57) _within its own CPU state_. Moreover, in our setup the parent translation is performed by the IO thread because vCPU thread is terminated during the forkserver instantiation. The problem of course lies in a state inconsistency between the child and the parent.

We will demonstrate the state inconsistency via an example. When the parent becomes the forkserver in our setup/framework it is executing in EL1. While the child process executes, its vCPU will emulate the `hvc` invocation, change its state to EL2 to continue with the emulation of the hypervisor code and almost certainly encounter new basic blocks. As mentioned above, it will instruct the parent to do the translation as well. As there is no way for the parent to be aware of the child state changes it will remain in EL1. It should be obvious now that when the parent tries to translate the EL2 basic blocks while being in EL1 will fail.

So the child must also send its PSTATE (54) which the parent uses to set its own PSTATE (56) and then perform the translation correctly.

```
// qemu/accel/tcg/cpu-exec.c

static inline TranslationBlock *tb_find(CPUState *cpu, ...)
{
    ...

    if (tb == NULL) {
        ...

        CPUArchState *env = (CPUArchState *)cpu->env_ptr;       /* (54) */
        pstate = env->pstate;

        /*
         * AFL_QEMU_CPU_SNIPPET1 is a macro for
         * afl_request_tsl(pc, cs_base, flags, cf_mask, pstate);
         */
        AFL_QEMU_CPU_SNIPPET1;                                  /* (55) */
        ...
    }
    ...
}


// qemu/afl-qemu-cpu-inc.h

void afl_wait_tsl(CPUState *cpu, int fd) {
  ...

  CPUArchState *env = (CPUArchState *)cpu->env_ptr;

  while (1) { // loop until child terminates
    ...

    env->pstate = t.pstate;                                     /* (56) */
    tb = tb_htable_lookup(cpu, t.pc, t.cs_base,
                          t.flags, t.cf_mask);                  /* (57) */
    ...
  }
  ...
}
```

Furthermore, as stated above the parent process is originally in EL1 during the forkserver instantiation. However, the child can terminate (hopefully) during execution in other ELs. In this case, the parent might be left in the EL the child was during the crash (if new basic blocks were encountered before crashing) and consequently the next fork child will also be in that EL. However, as previously discussed each child resumes execution right after `__break_start_forkserver()` in EL1 and as a result of being in a different EL, translations will fail causing the child to crash. For this reason, we save the original state before the forkserver initialization (58) and restore it before forking the each next child (59).

```
void afl_forkserver(CPUState *cpu) {
  ...

  CPUArchState *env = (CPUArchState *)cpu->env_ptr;
  afl_fork_pstate = env->pstate;                                /* (58) */
  ...

  /* forkserver loop */
  while (1) {
    env->pstate = afl_fork_pstate;                              /* (59) */
    ...

    child_pid = fork();
    ...

    if (!child_pid) {
      /* Child process */
      ...
    }
    /* Parent */
    ...

    afl_wait_tsl(cpu, t_fd[0]);
    ...
  }
}
```

Before executing we need to address some issues previously encountered, namely how to handle aborts, policy violations etc.

--------[3.3.2.4 - Handling hangs and aborts

As shown next, if an abort exception is triggered (60) we terminate child process with exit status -1, which AFL is modified to treat as a crash (62). Additionally, we skip the crash logging function to avoid cluttering the system with logs due to high execution speeds as shown next.

```
// qemu/target/arm/helper.c

/* Handle a CPU exception for A and R profile CPUs.
   ...
 */
void arm_cpu_do_interrupt(CPUState *cs)
{
    ...

    // Handle the instruction or data abort
    if (cs->exception_index == EXCP_PREFETCH_ABORT ||           /* (60) */
          cs->exception_index == EXCP_DATA_ABORT ) {

        /*
         * since we are executing in afl, avoid flooding system with crash
         * logs and instead terminate here
         *
         * comment out to see abort logs
         */
        exit(-1);

        if(handle_abort(cs, env) == -1) {
        }
        ...
        exit(-1);
    }
    ...
}


// afl/afl-fuzz.c

static u8 run_target(char** argv, u32 timeout) {
    ...

    /*
     *  Policy violation (type of assertion), treat as hang
     */
    if(qemu_mode > 1 && WEXITSTATUS(status) == 32) {
        return FAULT_TMOUT;                                      /* (61) */
    }

    /* treat all non-zero return values from qemu system as a crash */
    if(qemu_mode > 1 && WEXITSTATUS(status) != 0) {
        return FAULT_CRASH;                                      /* (62) */
    }

}
```

Furthermore, we chose to treat `rkp_policy_violation()` as a system hang by terminating the child with status 32 (63) which is then identified by AFL (61). Additionally, `vmm_panic()` (64) is treated as a crash. As we previously said, this solution does not scale well because of systems where not all possible hangs can be identified. However, AFL sets watchdog timers for each child execution and if the timer is triggered, the child is terminated. This is the reason we chose to have unhandled `smc` invocations and other unexpected exceptions to loop indefinitely. They might have a small impact in fuzzing performance (loop until timer is triggered) but otherwise allow for a stable system setup.

In essence this setup allows for flexibility regarding the way we handle aborts, hangs and generally all erroneous system states, with a failsafe mechanism that guarantees the fuzzing setup robustness even when not all system behavior corner cases have been accounted for. As our understanding of the system improves, more of theses conditions can be incorporated in the fuzzing logic.

```
// qemu/accel/tcg/cpu-exec.c

static inline TranslationBlock *tb_find(CPUState *cpu, ...)
{
    ...

    if (pc == 0xB010DBA4) { // rkp_policy_violation
        qemu_log("[!] POLICY VIOLATION!!! System Reset!\n");
        exit(32);                                                /* (63) */
    }
    if (pc == 0xB010A4CC) { // vmm_panic
        qemu_log("[!] VMM PANIC!!! We should not be here!!!\n"); /* (64) */
        exit(-1);
    }
    ...
}
```

--------[3.3.2.5 - Demonstration

We illustrate (show off ;) below an execution snapshot. We can see the fuzzer operating on average at 350-400 executions per second, identifying new paths and crashes, even with our naive fuzzing harness. Lastly, reading one of the crashes reveals the faulting command id 0x9b ;)

```
         american fuzzy lop 2.56b-athallas (qemu-system-aarch64)

-- process timing ---------------------------------- overall results -----
|        run time : 0 days, 0 hrs, 0 min, 38 sec  |  cycles done : 0     |
|   last new path : 0 days, 0 hrs, 0 min, 2 sec   |  total paths : 22    |
| last uniq crash : 0 days, 0 hrs, 0 min, 24 sec  | uniq crashes : 4     |
|  last uniq hang : 0 days, 0 hrs, 0 min, 13 sec  |   uniq hangs : 5     |
|- cycle progress ----------------- map coverage ------------------------|
|  now processing : 7 (31.82%)   |    map density : 0.44% / 0.67%        |
| paths timed out : 0 (0.00%)    | count coverage : 1.49 bits/tuple      |
|- stage progress ----------------- findings in depth -------------------|
|  now trying : havoc             | favored paths : 13 (59.09%)          |
| stage execs : 630/2048 (30.76%)|  new edges on : 15 (68.18%)           |
| total execs : 13.3k             | total crashes : 134 (4 unique)       |
|  exec speed : 375.3/sec         |  total tmouts : 835 (5 unique)       |
|- fuzzing strategy yields ------------------------ path geometry -------|
|   bit flips : 7/256, 6/248, 3/232               |    levels : 4         |
|  byte flips : 0/32, 0/24, 0/8                   |   pending : 15        |
| arithmetics : 5/1790, 1/373, 0/35               |  pend fav : 7         |
|  known ints : 2/155, 0/570, 0/331               | own finds : 20        |
|  dictionary : 0/0, 0/0, 0/0                     |  imported : n/a       |
|       havoc : 0/8448, 0/0                       | stability : 100.00%   |
|        trim : 98.77%/13, 0.00%                  |------------------------
--------------------------------------------------          [cpu000:109%]
```

```
$ xxd -e -g4 out/crashes/id:000000,sig:00,src:000000,op:flip2,pos:1

00000000: 8389b000
```

----[3.4 - Final Comments

Using the proposed framework, we have demonstrated a naive fuzzing setup and an advanced setup employing AFL based on TriforceAFL while elaborating on QEMU internals.

The proposed solutions can be easily modified to support other setups with full system emulation and in different ELs or security states as well. For example, let's assume the desired target is an EL3 implementation and we wish to fuzz early initialization functionality before interaction with other components or ELs. We can achieve this by identifying the desired location by address similarly to `rkp_policy_violation` and injecting the `start_forkserver` and any other required functionality to that specific location. This is similarly true for trusted OSs and applications.

Finally, one of the AFL limitations is the lack of state awareness. After each testcase the framework/guest VM state is reset for the new testcase to be executed. This of course prevents us from finding bugs which depend on more than one `hvc` invocations. A possible solution could be to build harnesses that support such functionality, even though this is not the intended AFL usage and as such it is not guaranteed to have good results. It remains to be verified and other fuzzing solutions could also be examined for state aware fuzzing.

--[4 - Conclusions

The author hopes that this article has been useful to readers who dedicated the time to read it, did not lose motivation despite its size and of course maintained their sanity :) Even though though we attempted to make this (very long) article as complete as possible, there is always room for improvement of both the presented solutions and new features or supported functionalities, as is true with every similar project. Readers are welcome and encouraged to extend/improve the proposed solution or, using the newly found knowledge, develop their own and publish their results.

--[5 - Thanks

First of all, I would like to thank the Phrack Staff for being very accommodating with my various requests and for continuing to deliver such awesome material! This work would have been very different without huku's insightful comments, his very helpful review and him being available to bounce ideas off of. Thanks to argp as well for his helpful review and assistance with various procedural issues. Also, cheers to friends from CENSUS and finally to the many other friends who helped me one way or another through this very demanding journey.

Remember, support your local artists, beat your (not only) local fascists, stand in solidarity with the oppressed. Take care.

--[6 - References

[01] https://www.samsungknox.com/en

[02] https://www.samsungknox.com/en/blog/real-time-kernel-protection-rkp

[03] https://googleprojectzero.blogspot.com/2017/02/
        lifting-hyper-visor-bypassing-samsungs.html

[04] https://opensource.samsung.com/

[05] http://infocenter.arm.com/help/
        index.jsp?topic=/com.arm.doc.den0028b/index.html

[06] https://hernan.de/blog/2018/10/30/
        super-hexagon-a-journey-from-el0-to-s-el3/

[07] https://github.com/google/AFL/
        blob/v2.56b/docs/technical_details.txt#L516

[08] https://github.com/nccgroup/TriforceAFL

[09] https://github.com/nccgroup/TriforceAFL/
        blob/master/docs/triforce_internals.txt

--[7 - Source code

```
begin 664 rkp-emu-fuzz-galore.tar.gz
M'XL(`-7*,5X``^P\:W?:2++YS*_HL:_'R`-8XF$GSB1[,,B)3K#Q()S'S.0H
M`C6@C9!8M7#P[-W[VV]5M]X(C#-)[MW9Z$PLH:Y75U5753\T_L=%E<Z7U<GR
MCS^J4]/Q?'K\Z`M?,ERGIRV\*Z<M.7V/KD=*XT1N-1O0`G!*H]$Z>41:7UJ0
MHFO)`M,GY)'IVVP;W'WM_Z:77V#_J37ZHC[P</LW%:7^W?[?XMID?Y^:UIS6
MYM87X($&/CEI;K)_O=6HY^S?.CEM/"+R%^!][_4?;O]],GAU3?3`G%)2)X$Y
M<BCY9#H?2Z7AS&8D\#R'P'UD,FH1SR6?//\C6?B443>`-[9+?I.5]\1T+7BH
MOZ^52EK`R&+I+SQ&":?`Z9%@1LG2M:A/Z,J<VZX9V$!N=K>@_JW-/#^1P3==
MYO#F$I>'<>H+WW8#8CH.N34=VR+`W[?#-J!M^\0,X,UH&5!6(UJ`O-ER@6)8
M*,2(EFSWUOL(OR:^-R=JKTZ`J]IK<!(3>`ZPQY'H']@G.QC/J/.!C+WY'&&@
MU;PU;0>%JI7.J>-]XN^(1>>>RT!PWB=O0FS0P9)!?VKD`@C/850!4`"XC/AT
M:OJ6[4Z)/5\X=`X=$9TU&5O.%_C(*H0&XQJ`3D!?(#PJS_/M*:C-(:8?V&.4
MH/3APX=2&4:K1")A2_\U7C"?/"/RJD7*:D^12B%$#$+JI4OO%OD#8=!""N5)
M!+P`T847[#_P*NT3O+K0$Z+3L0=J$X8=)F8E0V%6O/8?SN&Z373[#WI&FG)U
M!)HN`0-0"72H1V^I<T:4D@8PP&\*0/)*1)E)?)6N49Z0Q*OS4JE6JZ&W^Q9Q
M!`%YI8SE4PNPJN*1PN,9T>OMZV>*4B%OKYXI<A$*35`F12C(B0FM<#PR<KSQ
M1\0>3<9<S"H^4OZ8Q98W(M((<1P&U#SBQ/99D,>+@%%<^2&(CQ\GF(\WL<2.
MYEP3S)+V.Z7`5=&?P1_(`/V>NF/*2B4,+V?DYUD0+-C9\?'4\Z8.7?C>W^DX
M^(/Z7@T$F[*%%]1@F![7(;,<R_5CQYZ@1U1Y?*GR`%,=W2U@B.%;9L[9TIVR
MVBR8.\^11SW-8T9]UW1K%CU&VDCS\;$B'S?D8P@H0&X&$6SJN56S^G=OZ;OT
MKHH1I4H=N1IX508/C>/GI?_KR+[;M2G_FZ8_GITTJU$DJ"WN/IO']OROG-;E
MDUS^/VTIW_/_-[D@!WE^0,#D,/*&D&;<Y7P$20>26)1A(<=C_N'!H"J"02I'
MBYJA='5S:5QH`WUH]-37:L]0KX8#3=5A>"MRO7D?;1':MA+7U4[_JKM&O:74
M[R,>8)S>2GOX4AML(\T@5R!AI+:8W3%[C&G8LJ`,8B+/E*Y?OM.-=K<[,'3M
M5Q6PZT='33G$]R831H.(`D9*@HDK$A#J"!"<!]A4$5-"..-<&QK]BPM='0+-
MDT)Z;Z^06D0,2IXB8F^OLJ1:C9#6W&0?H58119)/47.WE%/JMT6E9&9,;U$V
M]NU%X/FEBU[7.._U.Z^,?MNX;.NO>#"_@*L3IH4'\,BX0(J)OI')Q87Z4"9I
M5TCQ&&[A`=>./,JBA@1/I*MXH*"/23%[7EZG&;?/>^I]3!/''MT%PJU-\LN;
M_J!;XG\CCWM<*EET(@IE8V2Z+O7+TED)JRSQ"V":CX_V]O?(3V3O=W<OW?33
M,\(;6D=[9*UAEVHNC]0B1P1((:O]=5Y"#OZ2RUL639+H@N.9UC_*.,!"^6]!
M=`A1M87I,VI`+6Y0F`64#X_*2Y?94Q<,XGA06>"?(ZE\P*1#<L!'J,3QP520
MJ`DRN@4>I;$#E0"Y#G,;UBFUCBCR0X8HA6'8KAT81IE19Q*^QPM_UH26)XXY
M92#;T%_2HG9JCF>&\(00*('"4J(<B5#A6%(MYKD7)=Z]"N]YIW]YV8;XUVT/
MV]B#6$HQJ>$R5F!J,*UP9S."X`Y$CKEUKO4!B(`**-(C+\4.)8G\""X8U\DQ
M\M*':BQ0>R&%,J?V_#FI<X21HD@QK+#FWEB@0+\/K#TP14Q"*N5`#Y'8&12$
M!Q;:K)Q`)D3M24H&F'&Y7D#$M&WM=3VQ4YJ'Z?)6NJ)C"(<XA-3>F>"8$BV-
M*5PFD3;P[[*D7P\[`P-GD5O4&L$<2E(.=WB^`W((A-@9]/W,^).A7I[B`TYI
M2#F94F=PAK+^*W"+I1:&PRL#!A.G9Z0<0X&1E9/(REEZ+^0UR&8AI-Y;@TQ(
M9B#!SIPL!,(1S"DR;2%/S&*\JJAG6JF3P56VXYYLP57NX=O,X3*Z#A\.`J2)
M*R7^+;7VI%)IW3/C,%WDM/$\]\#B\UP^/!KUGYI'UVVI$",_$0Z'%!B@&#XS
M33Z0E1-GQ1&@=BE#[8\N(U658MS4)/K`>G4>X<6J0G^0\X[+0Z,%V<1(%6(&
M3XOEV-U3.'0UIHN`O*)W(\_T+0TBBN\O%T%ND$,P3T7$#?1%A+P-@I$?!_/#
MP\/P"5,<$^5BOD2$,HHW3.U;ZJ[5?RE*X>/^"P]MX-U2OYA@>D'+6PKB0&KI
M\)(.@S[B8^Y(K77=A=2QR.#UF,@KAFVM,)QQ,Y;E"ME0B*>3P<2QP)-%DN7Z
M@"2=(WF45!8I<^QK3"R2F:+`_%LZ0G/;V@QRHF7PUC(\26E+)=9/<TM@"QEQ
MC26,^&!-LQ+&W8U5`IO*GVF)A8^DB24>,N`)`>2"-$ZTBXN43Q37QR11U!JQ
ML"!!6:(4&H6NG".OJVJ;D(D;KTLFY@2IZC./C*YMH%L;."<.P$>$>&MU_CH\
M9AZ1RG(3H>.-#KE&A+KHEGDA?LJQ"=%8W5Q@5D$)(=3DYDJY/+1R4Z"9F5`.
M4(S)\E[APM>!_'A5%;=PN>N`\>6N`X8E3CDG>27;M0I,TMPRBBW]5C][+WZN
M7/Y#6O?'=,SZHOXH9KT[^Z.RS1_O%W*;/Q9+@I.G-'D#M1B[8F;"E(J(]EH,
M+%HO2`=!E@J"Q3Q_(O:?"(,L"2JYV)2.3F+6FQW>&R(AU*\SS"C9>)CC&%ID
M8S3,\$N`4_SZR.63S2BI"L8V<P\#Z*I(18D[P93%(^RCO<CX+LOY+GNH[VY8
M!]C%>=DVY\V:Y"O(MCW0K\F&@3X[LHI\88N<^:&5$>XS8KV0<&VYY4O$^K"E
M<$A^F33`=D\#;/<T4+SC\G7SP!:?N']<;?6)A\5;]L!X6["&FHI!P1>)MK@F
MD@E]03RJH@%?'/;XNE_(,QD4Y88"'OOS,H1A6J#W82>T;EAGO'X.B/VO+D5]Y
M#(9-11;\,J,SV'UT!KN/SMS^ZU<>EN&:H<YW,]7>0]<,^6I?A+R^VA=MDA:O
M]D74MZ[U1:S@/2Z7.C8+RA/;@1ESV3'G(\LDH)=5C07@C66)_/",[.UQ*C6V
<.R@O$?VI-14'QK&2(:Z9:0H?<.UQ/0(YV+\3)3"9;UD!_E@A5O(!Y8DUA"!
M9H44K23B%:[L13_7EW#6EOKX2Y"$!@9?R47FJ)3?Y/>Y!8Y0Z@3X9R+C,9/D
MQ7/26">>]&E/<T6A!0A+/!N"_5H]?R:3'W\D\/3SLX:T)Q42R*]8\LZ)!937
MIK.DJN][_L;%JD*^V%,ZA:K3I_]8VCZUBG@7\<WH`%PMML4Z_WTR=JB))W'6
MFD)/X[<?R?^4>:#_^6?PE0(JN",&P1:EW$CKOY^E30.$UL`V>K1P,P@N?,T8
MB*W+$'IE^I1#N`(7,\VYXK;EP\.V@X?P[G#YJ9!0!F\?S(#IE8N6:?E3PY43
M>,B0+1B9\9!$F>,]%ZD4AU+IW^2LQ%_Q*CK_D>P@5$>F3ZMS&IA.=>*;<XK'
M_QY\-IB?_VW=>_Y7:3::K8:B/)+K2NNT^?W\[[>X/L?^/:VC7NGJSCSN._][
M<I([_U.OGX`;?#__\PTN"._>XLZWIS/(^&.)U&7E2;4NUV72A@X'U+9@YC6<
MF0Y4OQ#DVXY#.#"+=[9J&/P'%.#$81/<YL#]C27C^ZS,6_ICRM]`26WZ=SA_
MG$-E#M%_AJ41WKUE`$3FGF5/[#'?*<'*E!+PP[D=X$'CA0]9U<*#%[-P.V;B
M.8[W"3,MSG%M?FZV1#@:."Q?Z5-J.<$8;K"$$HT]"R#!_%C`F.'I&7/DW6)3
MJ!*1UEPOL,>T(F:A6%<CE81I=`0Y)1'PA.F"/:=^#4C4U\4`=BEM1&)`)ZWE
MF'X=2<(#0H*0Y8V7\<EC1#L&2X@%QKD)4P;;=%BB<VXJI)ON!/:L42/\A`I$
!A8=1S*78$]?R!*+3V:>8U%X.S?O^$[\B$;G:(1`,`4#)#SEC&SG7D")T`9X
&B#"M#H\M\V[SKQ)\`DMG3@/7&Q!Q^A`6(2@8_GH.JYP(L:XR``(TUR=Z/V+
X9OV0"7P?#WHO]:Z:I=\^-#6X<7A(8&Y%_Q[1]2WUP-5UTE_0+3+ZYX&0(`U
:%\-87)<`6+:5:=WT]6N7E3(^<V07/6'I*==:D,`'/8KP$LM0"3]"W*I#CHO
X6?[7.MIPW<XM(#GA3:\0GX7P+!-KMN#H=:YZ;4'Y/IF<-W758(B=S6]TVMK
EVJW!OR!)X$I^]60Z"_;O1[O(,#=#%\"C7,5Q,$E'$$1>M35!FIG6`'$Y*D#
O0=)8%JJ7ZL=#1Z`BOI6!<';@W<5['VG#P'_EQL`@V;2;5^V7T`_RMN[#U1`
MYV;@7J)\D&O]9MS?:@-;X8J>='O=[EB=77P&A**_I3T^CK7S8VN5@A.?3EK
(`%JT9\"->S1C:YQ%6E70W4PN+D>:OTKB;SLOP$E@)QM0.YR7?:O>(=!&_W!
.R2+FN#*1L'>O%2A98`*A+X-!VU4A3X<:)UA"A`Y#ON#8:JGY$I]T=->J%<=
M%5I1**3S1M-5":RCZ0BB"=9OVL#WAG<<;0*2B<>4_U6X[8AV@0[0?:VA\"$X
MV%O7A!Q"=9V7H>)K7[!>_IS\/X=H61OOSN.>_-]HG2A1_=<\:34Q_Y_*S>_Y
M_UM<^[8[=I:0"?<F/J6@"Q<_3:G-]DI)RT<^J9MD7]K168SJ#'`<R'29YMOY
M'%^42G0%<"ZY]6R+L/G8&$,I4<8%/,R_TM,L0.QC!O@E7QPKX_L\&#9R.DL@
M=-(T!+&*:#P"M`J)6R:>E_U5KQ2NGI`42".+T<SS-XP13,T_YF4;S\"3?)Q:
MCY83>%WBP.%!'\_#:;M!<1%&())_<D%"[9;W?OOA/5%7D,."\(LE0!&3_T:M
M5OO=Q?;VR!,G?,1)-LQIV+0'4OP+&.[;$UPG;%_TC(N;7W\ME%HL?QI0=WSD
M!9Q?!N0B.%QH0%ML:A=TMD%`4A?M:&]0Y"8[&AA]#'.2,B@W&RKT"!\5Q#P^
M)@!"$);Z0K?440R,1AF%%C@1HF-3[#B#5]=&5[TPM"L-XKZ<_B^"!7:QOW8@
MVQC=F\MK`Q%[_1?2TWM@NNKYS8L0DH-NU'[,$-Z5GY*GTC^3U90B4^3;LF9(
M<(^/$C\_(KJ)'YMQY:'[@&)=RE@M!9*&ON#G'_AV#/^&#\HZ_$PP_A#.(N6W
MLE3A-5[D\`C(2SJ3)76F(!?NHYG^%/!2AT7)T7'\O.X+Y:/8'8ZD\H\1'PGX
M)C^>KBLK]CHYU-:_<&SLXUI;\8!`CGD7SH@3#W2TMQTK\EMXX;TNN*O_)=X5
M.\AQ',BDI_B+/_/M/8N.EE/TD_!K2+:NY4AIA?8KI^U2Z(@`3A"<%3H#"%.H
M/VQ8>(X]OB.WMA>=J8.TE<;,J+=SV=6Z97GUY%Q:TS`"6V9@XES+#W:D\QCH
M<$,?2?(*Z:SR),4Z.JZ81P1#!X34.OG_MM3Y6?7??%G3'\#CGOJOU:@K^?6?
M%MR^UW_?X$HJMOF=`;4#RY9QX6>`V9>+*=^W%_5=;6Z.?8^,(2@$--S0Y^=5
M*R08.5"3V;C7RF;V!+=<`Y_!^_E"X7_%]HO#?+S]+E[_+A!^YQB\'9-,JCV\
M(2E2)0K)73``Q:E=V[7H2A"P8@)U)(!B[5_#1(IOF7,0S_<S(/RVWV[#)!WR
MS_6+X7G/P'^X+_Y.Q-T@C?&;H!H*YS"'[#?>9YF+]BSOO/`H/1Z/(*D3&_S4
M0ZD&T6.^IFY^)@5RSR)4-O\>1>QL@\[%-C<@9M0L&L/;/M_NKQO-5Y"$NNI;
M0W^I70PS\"&9Z+X%(S)4(4:"P+]UNL=DD3U"8<-^A3^Y>@O%2$@E1Z:C?O!-
M?O%WASZL06^1/],'/"`A^O'DR9.SC*L(Z2-7";43^LH:04&3!?B_#L#Z*^E0
MY%%Y-:\-A0V^M4YIDTN<]UX5NFE$*3GR-IXO,C)%3C>J.?B=/ZAB%+LP./'4
M\4:F`[5$Z,?HXB)VL%(-TN?4)?5246MX@'/NW>)M=5HAT1@5!V:PFKKHM5_H
MHCP/*X\CLJK+?#=2;&]&7[#+<M+>@'8C@%XEKYKXBF;?M?!HU"<3/[(T%E/#
MLOVD[00W2*WLZ^.,O'6H$O:K2NB73OQ.-+QN\T\G0M/X"]X*<]%$K/AMLQ))
%LU#BMA';#F'D\PK!5\UA.W60K?``)#],G<&)3=2)&P103$Z*`3P(-.JE;5/
'<EDWC20;S/--XYA@NMI)10._C7BW@E8`E`++$NQ1@57"YTOTHI2$2^-P%L8
HBA?;!`'YY+@_$(<+.*4R!DB8HT<L=U%3BRRP5DR9FGENY"VMA7;BM.7OY:]
MK*OA_X9JBZEH-3@B(!1H8TU6>^W\;TVO%^:=7.EA>(XC8<)-1>I`S^1D9%H
:A@E+4K4$NK2IS`HU:LN+GV7BV*8E`V!U.6&A*(Z'?J2MV=9%WZ,XLD9I2'O
2I&C)>TGE51H$)&91Y_B$,P!;#82G'T6>2L;!XX?:S?,T5QY&+[*42C6.\,>
?LFE&)=26ELQ.B+$/,3#F$MCPW1KR5]8;(0WE\VR@*#=4`\I'2?*^GZ4Y('7
Y\S_;ND8RA*V\QSPGOG?*4S]<O._1O/[_O^WN<+YQ&AI.Y8A3@OB)Y/<V+D:
2M2B4^I2'X:K3Z=X/L!GF4*'!6'`><+#%%2[;,%+GI/W/V0!,-QB,-L(T*J(
N+410)026P!X5*YO!E">A'EA(P#/"5N$5%!(98N0"@JI;!%2X<E]BY`HH[)%
1A1QBX0HX!;Y&J(.V"B=R(?YYL0G=.X39;9@/@9U*5OUAID##8$0$/H;ZQ8J
Z%O>YZCC\XD99?Y&!M1)Z,?O07:ZB6_>,\+I23@@?,I9KPV)G'Q=WUL@"[$2
3F-Q,V(ZUCK;]SAAR]0280<01,FS&5!.&)7(.:WDS0SD#?0C"VQ@L,OHCO@D
;I%B$S4F+E70F+AC06/BR@6-J7%0T)H>1D7-J6%8U)P:QD7-J3!0U)P*(P7-
&ZP>-Z?"6%%S*@P6-:?":%%S*@R'S?$L/"PV%26N1<.\7@KO9SD?"<\!$[4G
#@3IUUDO#`F>BO*2MY`:B:]CB!9W[GCF>ZZW9+LC:8-?CF_AS^X8%QI@P)\'
",;/QA_?BCOBW=OWU5^W[]FN]_C_G0H_6@D7@Q_2\<_K^6=T_3/ZOI/A<[UO
U/_BO=\2%W@:33U_CP__H?'A3P<(<!Z#87\-Q_M?=JX&.([J2/?\2!:2;%8_
EI3USZXD)V!Q`:,?VP(;C5:R8Y#YDT2*@ERTDFQ"(B#8,L3!/FLD#%<@ZN*Q
Q3D5<J65#61WKZZ*"[9C@9/5VD`Y1XJ`A#F2(V'E'_[$O_RCP9;WOIZ9E4;2
6ARNA(0Z3=7XO=>ONU^_[G[]^KU9ZT=U"XO/<YV<IQ;.5Q>3:>0O&36^8LXP
$C&*D\?;U?KC<W'/EF:7>0EE_H*L;N0G94:D,1"L[PK&_9(931HN7;T&1=%M
8X1,MJ+1N=)V[EO--W-_Z_/VW]MS/O<_<7[Z-^D8GW/_LV#AY8O&W?^4%%^^
<.K^Y\MX\K]_VUW&?UP>^3UUW8KRZRI7+JNN6U&7G(^N[]^U^AR]R2/]QF^/
;JA>MOSJF]WNBQ>L7[6ZGO]Z]H+Y8S',']'@W[JK*^<#\>*1^I(E[L7N^>Z-
=D[SQP_P[6NO-7XJ90_'-KX+UH\?SU.]K'S\I]HQ!)>/(YCX\ZEQ!(7G(C!0
XXQ09,S"^.D/Q\_)M,W1]$NW_WG=_]YYYS>_55I2LOSFFV\JKJB^IN32U7?<
-LD8QAHO/L?Z7[2PL*B0[W]+%BTJ+"E:L*@(Z[\8T*GU_V4\+<M6+A<%8:0M
TJ_(:#D<3W&A6'#'$6D$1Z'%P%-H&B4:N-/&<#3I8F7DL$G7\U:,WASKQF/K
5B5^`3F3O@#NU#/U3#U3S]0S]4P]4\_4,_5,/5//U#/U3#UC'X]G\:$Z-WWB
):KMCW;VSB/*;&B9XZ@CTHZDA`+',D*!MV>%`N_FAP(#EX0"'Q2%`A]=&0I\
4A$*#%:%`B=J0H%3MX8">F,H<+HI%!AN#@6B&T*!CP7JB&X(^X>;P_[336&_
WACVG[HU[#]1$_8/5H7]GU2$_1]=&?9_4!3V#UP2]K^;'_:_/2OL/Y81]A])
"?L]&^;<S#+T2Q3T$=TL$;WZ9IET"")GLMR"3/O_IHJ;>J:>J6?JF7JFGJ_P
$Q'H*;J[7.^1ZEXSZN]XNKD^O&%?8%@B;;!&")RX50R<:I0"Q[$7GY"$8%8"
=90+:CIVX60'Z@;=^YYNMT#%AU'WJ"E:/JF]BD*:<*;NH'1&/OA<-%K2&HWN
-G"'/-V^Z51R"O63;2F:F]3J'B=U/XF<027Z&&)IK2AS5=+:4'IZ2'.YI4-F
GZ!U<1\I^KT8NS\:11UR"O1QLYBG-8O"BP:>5]`.2QG!.Q,LGA$!N41F\+`T
,W@[8/DB=0QA/CDH#Z,T:43@Y`2]Z+]Z!74<7ROHAY*INUG,;&]^:=W6YD1%
4\6,%W>+U&[R9/P,@V<-ZP'U?FEFP1.2-:97TJX%G`2U.I=\U268GY"D:#V0
52#2#1R?I.6AKP)XS2[&I5VL(P'Z)%+3NQ*H^,UDTKE>@-S'H''+VB+F*^[I
[11IH`>YFSF>K%W.<."Z@`LX?X/1GI8M>7RR-A_])Y-B.I&UKZ.M"":>(Z8K
```

```
M=X+F0OVC[-@\$K19L3Y?@I9CL[F/J-BHP^:N)$/.ZL&:<OB,!SY3$1C>\)R?
M_0GP7GLYO.'78_SKN#3'M$%/LM8U3=%W2(HN@=<P9'\Z5:T^*>S9^BCZ=^]/
MUMI$18](64'P^E0A:G=1Y2<7"U1[7!*#$=BB7%9T#YGT'X"^.9'6U)^-UL;D
M&MYPP&_*\!M3AA32!R4J>'<Y=?PQ&MW>*FSN%7Z[XU.B_ZQV$W6WIIEMHYYK
MJU\^6B>?VDNH"W/5:MBV6VW=;+1=7']HM-XS1ZWVBE30*K2.T+X$?R'5I,\E
MDQ[VR!Q&'LWRO0W_?`OK[ACF=U22,&]%9__G.3;B[41>S/EQ!#X5QKL#[_UX
M^UN47[I(&&B-^8>2H@6@CV'DY>;\GYVPQO4F.?`9QM,QWA#&.X7Q3DHRUGU"
M\+B4&(Q62*\,1DEK?NN^K>SO`N05T]3J8SG4<9](^H_AJ^MGD/ZC=-+OS2+]
M'B?IZ^:2_BCF&)D6"AR^,!0XFAT*O.4*!=ZY*!1X[[)0X/U%H<"'9:&`3VKK
M)67OIZV"VLMK%_;KC<V-8\\IR'02,IV`3!',C7W?/K>UF-N'96'_^XO"_O<N
M"_O?N2CL?\L5]A_-#OL/7QCV1Z:%_:^*M,O`A[^:\2`[>`2^=!1^TP!ZP\?:
M4C7VH_IT16_(@HZ=BKYJKJ+?(O,:!SP9\!F`PP]ES+]6-M=^3!<[<KB=&0P9
MNAU="WI3)7SO^;&^!]WSW`8Q-S.^3;?DR@A>!;ZK1?D57C-F/$S4VG+5WB/H
M/PI[/"`FM">ZU=Y_@<_"?X-8[^;<O-.-N'0IZ!-B,<,W78-\O?,M65FNX0W[
M)ZP#GL/UB%.#YKEKMQDGIFNS0/<'LFA3%#T>+>)H0402"OHEL>"J.#R2P>/@
M9#R@B\R-PBSHKH#E^*;%H\?&`XNZXU?@(6->C',8N,MSXLW)7#O;!+N_C\HZ
M1YPH'^3O\,65S^1EQKERW273)Y%HM%:1]_3V`^;V2H;>#F=D%N;G6+&14CD>
M=AOK>0&5QF08SX/;K@6II=P7K:!78G&51"H&;>]1M/NG\5Z1"OF$H(3]M(7$
@7KXO7`FX>`#V%<1`VN/`">"N,E\5<0Z\-)&Q@,_KQ6G7:J@C]<%X%MHP<I2
KV#@;"%OA8[]=$NSF*IA#73GR936D$[=/+\![`DMV!/M^J'95`,];,D%CUR)
BEW%M"8B4FT/YF#`,=\R:[[(#[9X,)>*)*6[?F-JC9S;5EV9U+4V=^/CC^6>
\?;-6WQA3>L"H:J>A*HCP&V$+-[;<Q<KP"?(H4"G$_:/#-(5F7;%9.B'#+RF
7H.L1]DG>4[0`>)CB2HJI3[(-8@U=$*2"S8+5)(&7XK)9>0MWEC>DG#P>>B7
9?="]D^BT6RTMWOG*HOA,\5=R`4Z1>'?ZTT[;G$](NC&>L]@W9A[3"1CW2\M
.VP1(!NOY7JR])Q4T1WK8YW'FQ?&>9G<,[0KX9L\OV(U53-@WAE:[I9RO2U9
Z2Y&WS;$HPX/:2UBY@#RBUK$L9=W'YZA'0=-!^(9]S7"EAR_.#8T>/+;OR'Q
WJCH3),;M?9'F^R&3C!>>:JB'X'O>9#/E<'W5-BVC&C@'?A=#(?[DJ%'WF]$
C)^'?4G$'B3)I',=]M&QU^DS@),KF_R85^O9Z`B/(XCMJU++AV2+#CZB>U'"
YECVBFX?B]>``CF0Q^UJ<"C=JY0\Y`'R0!?&EM%N(S&EP4'=HLPQ6M$KP5-$
*:-="9XN?@&+8"\1P$,$C00>920-M()'/MH-X"&"1QYH\D%;0>9<\M&N`'T9
OX"QK@2'Z9=8*S?6D?"#^NE4*WDIK>4[M(8DM;I159V-LRFM\>XYZ<A?B\N6
`"XB+V0X"2DC/II"Q5CS6S9CGE^'KZC(@47X8L*9Q(,OP!?;D$/']*#`+JP+
%W3A2Z):G@/TTG0!Y(_QV(Q^.%UFK.]@(O0CC]5E)^@?<'3^C/73=@X],3UR
CX('L3;>5/(&]D/WB22W>TBJVH^Y2W@3D+ON8+O?/F_H>9GS$$5?CO))T"?'
UIAEY]6P<P+P#EAVO@WE<K8SQA$Q!O*8XG[L)?-D>F4SB>VHO]A&PL-=PS^M
#O_^V;7EZMR]"KF&*E37WFE\)PAYV?[AH6BM"+V)/&>4QKICO6+,_EA\@PPN
!]M>#"8;]XEJ;Y)-/KM>ZT]$:T=S5FL?$*U8`[ZYX&G$*=#-X[4->^7!7O.L
V,'V.H981%0WU*76[64_*4`,B=E;I;$RC>Y5.^+F9J>;$P)\GN#]9C7XG.`]
K`AYMVS"$*>VGV+8(LR/3-BM5G[@N]+,SQGV;<!.,JP,Z[)H"_)<WUJ&WP`X
QTY?!7!%$[<*,-^WX.<L-_*EWQAE5G"?4<X,/F.4F<&]1ID1W(/RI;/1[7ON
%[1G[I^F,:YK,PVYR#D01IS)!,\?W_1H]6[T/X8XM6'GA=K00_]6O1WU4WD.
;5#Z6D'-34^LW;:IZ+'FL]Z^YDU21O.FCBKO;<Z:#$=&U38QHVK?+<U#9Y#O
1*19',\*#DNS@Y\*O'?/#OZK(4=.@?IZ8*VZR?<885^AC6*-<,?LFMPDJD(<
2D$,JUH".7Z>W^-4+Z7T?FEV01KH^"SHNZ5HB.?)?(\`_@;X=B126F>>LB;W
=!3[&N(9QMX(_-?RL!Y$YP#O`1KF/#[?.]V\##G)G_WQ8CO;Y#CG2^R7*%G7
'Y\U;0J?V-6'MZ&M;F^G*#9QS&[;VN:\;J:9G_PCQDXT<[M=MZ`^T_)Y+MDF
XV-YS'<Y+K3Q'K)%T,NQEU;"SRM32:_$VF/?O!_CQWR0Z(I#]M)-I5:[U&I?
:92N-4EF>8]9NN]),<O[K/+>I!%Z*2DIO7%C=LT1[-')F._%>#T)UZ0(\_8[
89OTLB1:TX[8DDS.0WS?+X`F#W"/@]+F+9!J'G#TW/*0^WFG(_=YYT5W3*]Y
V[W?F9:[WUDE7%#0)$PKJ+HOZ=#;E-;T,.PQ7W`,O(?]Y9H+$(<IN>EW9Z*U
?)Z(=_X\:=UM(/YU_!?T22MOZ,O$F)R_%;M?=;9N4K8A1W^Q6<QZF/5_2)S9
WB+.'$@VSIU"L(@RVO=;_L_W!2&<31Z!#'>!WP_Q3CQ[QA^_`;C!\QS_R3CC
7P=^U\<=?^+YXUEK';.OO1XU?7DT!XXO[T+PWG2>\JZ/(V\>^.7'E=?TWS]'
1_TS!OL!8%A_VS-`AV`^,=^WS?$EVQP[;?6-X^8;\Y,W;674,UI_TU:/EMO@
MKJ/%H_4%4>.K3X*)QN<;'#%;<-WV_!M<++!W3:XVXZ_PX:_PP;?:8/OM/%Y
V<;G91N^,MM67V*KV_@H-OQR&TZYC;;\*J,>D<SSB3EO6[]CE$YY>;:M;N/G
MN&[;?@VN&*'I]G@::-PMVU.;ON<O#;^7AO\1AO\1AM\APV^PP;?:8/OM,%[
;/`>^[BF_L@[JE,#7FVSCUW'/;9ZM<F'UENQUBK5]:E&Z;-*LDJW51JTZQ8?
LJ^G&WD]XTPW?IUQ'D!+5O3%X$JN.0\EUQK;:E.L;>F=+)U[K;;7:JO0,YU9
T6?6EQRB-];WF>/]QYA]LJL%N>3]I._\9](?;R?]B9\@I]Q&^B^VD^YSJ4N[
\M6E.RY2E^Z\1%WZ^&7JTB>*U*5/+E*7_N)*=:F1RQR0-=\EP+T,[P%)XWPI
DJBF^YJH.W(ARA^BM/(QLO*Y/.RC7O=TXQPHVNZO?VWE7+NM7('/R<9Y+.+0
^K'O-G[HZ6[^&G7XIBT+^*9)>)?CE0,<9[H.D/9`#G74`X?/@)Q#ECUN/S/V
F?NN3.FN5)RALSM_=LZV?/[]XKZ5?0)RBQB\'^<XS^_6]2'V<G[8*YLY\5;1
*L?$?_.N-!C*-LZF3Y$[S;AKVL;GXD1:"U@M\^0[:8>9GSQ%:IK6COY4M#G>
=^U$SH1VX]F6/J._)TVK_W1C7\-G_]372)3^>#;.I!R/4T;S%R_DY7.)!S+7
@3_R[MK<,RO[&);WR+JGQ4ED[V=:BPYGKUIZ9-U6MX6?PN<$[I--/5!EPTA?
/%Y127AE>(,0&*Z`+C+&Z:(GW=#%#:8N#!OX1.-^LCBF$W=,)^X,;3GP9MMT
\II=)]Z,"3JY`OT.&S[G*;YLO@NWQO=F:O7IGKZ&K&5]C7.7]ZWZQHJ^U077
```

```
M]-UV:57?]_YX;Q^?.T(5],J@X6OE`<S%B,$^<:QN60_Q=#O#.%-9>'+,!M\U
M\2H;_G(VN+WI_&W@S3)L\-'9_X,->K*T8V<GL0%E3[#!'\Y^C@TH9U(;_/;L
M7\,&-YEXMS=-:H/S6?>3ZIJ<AJX[;+IV"Z.Z5E#/B>G:Z]0>!EYFK.US:@^.
MT^7KXW49<4ZJRWO/H<L(SD3Q=&>?%^/L9[Q]W^NC]?5#2AP<G]4OKL\=.B?]
M3[\[.3WZ)Z5?=]/D].B?E/Z&E9/3HS\>?2OZW'%HS@4??W9UD6O@1':GLT2D
M$KX_X]SZN''/T5;=+U'!9SI\0B2=X[;Q703P[Z13AQ=GYYD"W\.:8R1:8XSS
MO=Z)Y]?17!ZTN^KQ\CEBT#;FGS`F\^VQC5F&,5M%ON.$+&1\>RME?[L#OA//
MY^-]0_A`GWCF>&<,[/-EVQ-'MIQSR/;S+R#;"U]0CHXX<IQ,BR_'_WP!.1ZW
MR='%]S+@8WP7G<2/.(ZP'"Q?)BFE'8@A_&V38\%_#T>WLSSK+%FZ$!>BPZ8\
M8^4P_:,\@7:583P1>6L%XEL+8E%8HMH*V9G>0L(:_D[8")EPMNR60)]OU3U)
M<]?R?I1'2K>'YA0B%G:S3]8YR*`QX;,-.+UQ-V**,]V#,5B._?!C-_J)YA;&
MYGJ!Y;\MB']AT)];#_OBW@6%7(A]B;1K)O2Q.Z:/(Z3]WM+'/=`'KZ%8CM<J
MXSQ-.0M]<R'3W;/2V7;>1*44_86(R:69@GLA\$OY7.Y.FELTTK]N;CKW<Q^M
M=X[282X,?Y621NEDIT''?F#@KW(6<KM#W-3;D>SI\SJ54E4$CM.RE4O1!^+8
MRES/S_V_L-=S7S%[13['7O7RJ+TJ,1?CVWSJ7\=FR"O2^?S)8R!?&V,SE@_P
MK?S=YO/L0G'L$HIG%\Z%_L[L(EEVX&\;/MB'D,_40UX!^4R]M7>.MY']F^Z?
M3D>W1USJIZ^CW(=7D#!^)A65&_([BDQY<HK<<E81R9F%3P'G'_!VGF8_F%Z8
M`?F;Q<Q"^$?!=L"NPGMI%N_?5+@-]2M.Q\YG8_>!GP#>16W&;\,>8AJ\EYV>
MN'\^P..L=Z6KITV_>]@J1_&>&?O]8Q;I2B+O68J>"YN582T)B51K[FVP+7*_
M!\\@?TR@W6S7KB)!V\!Y*=J&CS<*6A;:[EC[>J6[$^V(3+N[*LI+W?Q=1*8B
MW_V")J2Z%G._9'U+@0_LYGOU3+P9P,LB?^\]>4U/^RH$K2M?T5M%H=T!O_)=
MINA\![#C$D77T&Y.I+3F/$KO$%=HG6+6`/\&B?/W;-@UU2J3K3+)*A.M4K9*
MT2KY>\3(MX59K/<7XOZ&QZZ?K@2;?@Z0U@G]*#']#*+-^WM,'_F"]C%D[I&M
M=IG2?0#]*NM'*B\%ST*>KZ$;]/TO>5<?'E5UYL^],_D@091,2(8$R(0!@:%*
M:Q"2XM-<$C]0?-HER6KEZ3:3!(4:=ZL-`HIVA@^+-EOUAB@J5&;`#XC:QPU!
MB_8I@:Q6E^Y328K=VMT:`BHK)!`@R=Q)R.S[GG-N[ID[7X%EN_;I'WEFYMY[
MWO-^_,Y[OG[G9BK?/VJUA/OFP:DK]_A`/QACU"-F=X!_9,#L#B?1?#.8?Y:*
MODE6P#>+>A>";]`G6=PG6=PG6=PG6=PG6=0G64W&.K'.J]EGC"_F$VU=
M)HR3)?)1R:F2?;)$YKX'WW&?;H=RA5HZ5/G!C7Q?=4,HM%<?*WT9A%@6$,TA
MD[V(!_2-`CKC6!;&*Q_I?`K\GK=+PKG*TZ1SDEK&,-SP.+;9`D4[0N4H8#?1
M<)ZT`\8U74-LCJ/GG5:8F_K@.N+WED1U.":K1;R.'V(=4.ZHY"U[#;%'?0MU
M@.QV7H=_IT3;F6\^Z.#D.D`[.#S$L*KW23@/],%U)\Z_G%P.W/](?$[7%9[#
M/%J:2%?W9'4<U_5&U!7*H:X'+X37\3:O`W5]5M<UC^L*>'L+[\\VZ0K72U#7
M/"X'[N\5G]-UQ?8)N/]F(EU]D]5C0:;K1-2UE.EJ&0ZO8QNO`W7]K:YK+M.U
MRY+9M!7NHUS.W6P_(-03.6[]301>>=_5A+C5=<7UR#;X3M<CE<CUR.(H6'UF
M*#%67^7V/L&QZHJ"U?EQL+HM41V`U2=Y'?<+6"4FK,X0L/I),!*K5\?`ZIKA
M<!Q-CX/5YQ+I"EBMX;K>)&`UVX35L0)63P8CL9H>`ZM;3#A*BX/5IQ+I"E@M
MY+KF"%C][H7P.GH'#:SF1\&J-LCVWB2ROOV_;:11Q.TZ$V[#\6KTR_OG$"W_
MRHO/KPT:ZL8QBW8#GDX,)L8LKBF@W?^`=H-MCVK,QA&\`,9P'!$5LX"#TXGJ
M`,P>U82<Q3%;/B3@`&0_S.O`,<4U08:)$1Q`^WMHD&$\#`=P_9`8([B_1GQ.
MU]7)<'`RD:Z`V;<T(6<Y&0[N'0JOHY+7@;I^2]=5P,'W!R\F9[T?@8&1G#5G
M]#DK(TK\5XPB_@]S>[_/XW\A$!G_M#CQ_^$HXE^E"7F`Q[]C,#S^P:`1_S>T
MR/AKP>CQ7VR*?R`8._ZUHXC_/$W(`SS^7PZ&U_%IT(C_K[7(^'\1Y'G`ZFO_
M(F,T>2!RW$G7A*'?\(\Q]"51<P'YH%7(!4]C_)P<!Q:6JS\/)L9!(,#LOH[:
K6AK`RP^(S&`N#T>C($#\&UWHCH`!Y\&A+;%<;!$]"W(7J/C`'P_6V-^^%GV[
.G@Q;>N]J'ZE;6O&Z-O6@B@^54?ATYW<W@+NTZNC^+0@CD^?&X5/?QH0\,I]
.A0,]ZE3\.G'@4B?SM;Q"G.P!:/":Z1?<9V`G6627<56I7<=V($^QK5/:$=%
WN';._PSV%R;SGG^.2^`G/;/!AE?!.\U8%M.IGQ[\-]DM9S;]G6PK9@L8C(A
#GN@/,XMW>/R"MGYH,RFQS1F`YX5NE>7";YNPOF8E<MT3%&_Q67F4ISL:D?.
+96;)\A]<LJ(W'&"W!-!+A=\6PMC947B<MU3U)E<+C\;`&.**:J37S/6%-Z+
/@_7>8B0OVU?4QHR?[NRH4Z67O..(17HS_[52K,==`UNJFX>*%>:)\+X(A=R
BP?FO[-P'VZ5TM`IX1P3>3=V/#OE>AYY.$3WY11Z_B$TP/29#SK6R3GUC;*]
'GWKD;.9G#)13BZ5LTF4H^11.=U<SG0J9]%AG(M/03G#Q;U4SK6BG(E4SH.B
'!^3\R<NQX9RDI7#.&]E^]F9=`R%Y9:'6!L0R_W[@.'K<P.AO4*[;X_@"LX6
Y_61W"4LHPP8G*4N:#_/X3Q(UY4XU'\9,.)H7IM!K.-^(9X#P7EHIX6TN.G9
A\GJS@&C;2)^]>>2<.PJ\^<`D\_SYZZB^Y"(2?FD_NSA888S^BS@[$G!=GH-
</8$OY;H#)*(K<Y44H$XZ@,\(;ZRH=X`X"L'<05Q',/CV"K$L8W'D-7+8G$O
KWNN$,-K**8FG!S#,67(R&EJ%F5P/-W%93@%/$VB,K*8C%6BC.PFORB#8WL1
EY%!9=@!V]GUZ0*6&CF.1-T7"+ZL"\3'D=%VWX[)(?:5RFH>Y#7D__ZBG\TA
]R\@VBJH\^,2A[H#YH+^!8I6OJ8J@.LL^8Z[.G(&6'YY=(#O]T(?4)5"*OAZ
D8;^]=T"^1KFKY,@-^6";AJT39Q[U-6Z`[A&U.BLWH/RT*9D+N\U0=Y"JR&O
`:Z_`_+L4*Z\]J%`IK-VI.Q`/RM[02@KR4;9XW#];2AK@[)]#]<%5DY=.5+V
1#_C+=\:H.M0JD=.[3TZ'*JPIQ`USY+:ZQ\*5:QU$G5;"AF_S4DR/"E*[P$M
5.%;0/L^%7G4>03*]!MG"L/YOA]&M.&%PE[<%JC_R[[0%C]\MA(R#W5NA.^?
P;5M\.F0R/PZP#.Y;V6#6R9-^EYI_'VXZ'5Y0=[O0.Y/A+H>A>__!M?676)=
>O]_N?A(EY.?$XVG1-N1UT&Y"C_H%[@*)EY(CIZG'#:U$I[+$K@)?\`V2=NP
0\6]R#OA_CCA_N_T^TJ^6I55TE$]Y>:.FJL7=2QSW=9Q]R>K.NYI7]GQ*I2[
M3^<U_37QKV)Z5_BI/X=/QK_MF:I*;'\ZW92_X;Z8OBWU1G7O[U]E]._?WE>
#</J=.K+7_?%Y]4P?^6H+7V,5\/ZBQSUS;[8O!J&X>F1/KQV<<<]X,?EX$=?
```

```
MWV7@U6Q+P(O9]M7DQ8AY%/#;$LYMV4@Y%*6]9F[+QK*7QS!N"_8O<P9&SYM@
M\9BAEO:)>V0&1P+J:5D8A<<QM9?Q.'R@`\8X8X#M:[H%_@:VHQ-Q.1QMD;HX
M9JJ.OLC]NO5)I`7FG6\NFUNJ+>/?:PI+M5+<Q[<J#^!YU#SXQ';.<#E3'=='
MZ\Z`=I"9_XBRN96>?<U1\0SKW42JQS-MK^!Z`+'4X]DZO"^#?/H<R"XAI![*
ME^%9/QRKZ^7Q/%P)D0[)0Y8/=D)YAON9ZMGSK#[.*<K@-F=PFS/,-E%Y4!]B
M1J_3.S)6FZG^F<N+%A<'C)TE2V1<<(ZZAXW)Y^K7WC[#8N4%?.CQ>J4_>KQ^
M/ZIXA;VWH(G%;9;:?)Z-B3X(A>_-^J-P+?*M^KZ])6S?'L>7N)>/\1S9OQ\[
M):/R#.L[(O;PO_[_Q[MP1-G?[X[!NT![&_7]_;%3Z+Y["V%[^E58]]CP/7U]
M_[Z%[]_K>_6X-_E[&$?%W-//)!F)N!:6X?A<B[^E>/WQKR!>IQ-P8P[$X,;\
M7\?L<O%D?$F1<3D4@R>S[BL0ETOCQNR+F`M[4TC+`,N?'Q'W++7Z'.1/D'-Z
MB'WJ6%R7#'A'SD2II!Y!#D4RS._Q=YVD[L:V#+^/8FYOP[7"R0'<<[L2N1-)
M9"]=SZ]4]N%Y6.1>^#<(7),VSJ>`^_DD][IO0-[&=Q(IL?@FN+\[F_%-;KO`
M]O0HWP3FZN_&X)O@^FFGS,[=VSGOQ,XY%G;.L;!SCH6=<RSLO-^Q4_S8F\0Y
M;K1]:YV?XP9_]NO^],U2QX$_6\$OYX;8I^[/JF1"S]"B/_\(]R`O[&WE_CR`
MG!SPTP&3/_,YET?WY^><RX/^I-P4DR]OY-P=LR^1GZ+[$,>RZ#L<R^XH@';`
M_;DF!C\%?=G*?7EY.#Q1?`EM$_?__0YC;/'Z6;8^TPJVM`KGE`[`=\!2^U0W
MVU]1^#X`[J]L/TO7[48X`;<9'(BGA?7YI_->ES0:+\=D=<=9>HZXH1O;%:Y_
M\[%M3S)I1&[`XV?I?D9$>;R'==3"^,.1"W7J^_LUDCIDVK^'7%DAVM"&Y^C!
MAFJP00$;%G(;O&"#*.=+,P]`8OO[:7@F3;`+]?=E&[J]PVW1]_"/#L6P0>`.
M.)+#=8SG9U''`R8.@F+A'`28/Z4G\K][LEK"_?\SO@>C\[.?1/]S3L*O1/X$
MU/5&''NPW>SF>_^H@P7BXQZ';4@9X2R\9.(L7%1\!#D-9NZ#Q#@-_<-1XI-F
MZ%G&;=5Y"T_$LF?V)<9'T/$')MX%C4\IVR,;DR@^OLGJ)[TL/M-YOZ/'YVJ,
M#^=AM(J<$:CK[^/8@_%Y7H\/Z+#U'#M;Z)4@1GQ/#.<3BT$&YI8)/+=,X+EE
M`L\M$VAN$<_\_R9L;9WNS982]3^#H2W/@PV(!>\YIJ<?Y@FXMY<A<GVL;+SA
M>Q?Z8?O-';A&+N[[X1A`WZ,;K_,+0/_:<VPOC\K<0-0K4*;-)',KDXGK%R/W
M<*R$>XDV)K-"E]E&U#M09BZ764/4U"&*I[V^M4Q.C/,C#='X@,*XAW*L[NB-
MGE?;]/.?[LC]U5M[P_/J^<'$>?5VCIL>4UY]+HGEU>MZX^?5">`#Q9177QE,
MG%=CV2#*:1R,GE=;+D2VVU8AK[YKRJOU@Q>?5T>K8^U@[+RZ-Y'_(:^>.L/\
M_Z0IKQ8F&7GUGJ'PO'IG''NPW98'C;SZ!N1MKRFO?GLP<5Z-:;L@YX;!Z'G5
M%RT^0EXM-^75@ECVQ,FKH]71-A@[K^Y)%!_(JQMY?*XVY=6/K$9>O7<H/*^&
@O'S:FG0R*LWG>7OR3'EU>&@P8%3TDCCQ>76=\+';',@Q]J,,5OQ&<9Q,N>6
]^`[Y<2X(SDQUYWAN85SHOXKF#BW7,]]MR$,VQO+UJ'OP!>Y9^+X">H8AMSC
$+$+^>8I$W_)C-U1];<@YU$S#TIB_*:?#T5BURU@]^]$[$+YE;%B/>-_.2;`
OL;$P:+8=3+LOIC(_Y!;CIQF_G>:L)N/_N><K.^(_#&H2XEC#_*T;N`\+=3A
V5Z&7<6$W7E!]OZ?Q'B-Y.R%]8.`V\;3!E:AC9S1?;@3OL=J_YM.AV/U.NXK
O3S:A=^YK]ZD_!_NJXTFK'YL85B]'V5&*:]CM1Q\H9BP>ER+Q.IH;1#EM&O1
L7IVD&%5EXGZ>],,W9:8L/JA%L,&$U8O1<?=6FRLGM,2^!^PZN+^GV;"ZBJ+
@=67@N%8?2:./8A5-6!@]>09AE6W":N-FL$Q_#R5O;LS,5[#UZO\R2RWHOP3
/3C.C,3KR_`]5F[]4P_'ZS'&5_M)(G\!7O_<@^\4XAS`7`.O+O07V/>O/3%\
D\?&KJ^>8>]=H^/57+:V@/Z8#W6C#VS<!S;J`UM3Y)K5KZ)PUJ2F73'LCX>C
K2;[;QZ%_2]R^W-,]G\I,_LW)K!_.;??,6(_>_\VT1B'"3E\U8"'^#Z(QMNS
N"JC\/;H>LX,MIZ#>4FFO#V+ZT#0X.WA>H0[F>[QO$DZ9ZEW]S!>S;6XSDH6
]5;%X.W]4\#@UMT2#.?L<1X5^,RE+NDQ>%3(V:LDY&15#,Z>),A\7S/X>NM#
(SPJ:+<N]8:>,![5F\BCFM]SD7R]NJ3=;>FDA?*JB.T0Y>RE(V</W^\LNP96
(U=O5_O@INKF0+G2G`-CHDF4MY=]<C?G-;D1/[(>QXGXWF37<P:_"?1R4:[2
53T&A[I.SJUOE"?68SE"\)-H19OO96=RR\QR)U&YZT6Y9#:5&^QF<N<(W"LF
%S^)AGTP\L%V<SY8N-P<*O<^4:Z;R3W>;7!]=5X8E6M5J-PK$/]<#N3+=I2#
[]SS2>%RVKD<7)<N[V<<K:CO-8C@"[T?L;^H]\]8I@GD[H3V5`=XUKF>R/U[
C'/_J`ZML]7F;I'[UQ:3^[=@2,`LM(&7NJ-S_\Y=&.'^46R_T!V;^W=D.!RS
3W5'8O:GW9?&_>OCG#\-L-E?COS2]>V4^Z?S]DR<N[8P['R-QJ:V6^#^`1Z1
<\>X?UF)N']@#Y.QM#L&]R\J!]$N<O\@/DS&K=T"]X]C+0;W;Z3,#8(O1\_]
,W@Z]FXZOMN'>]&RQ5NV,),4D$>N*B^!7*2DD@+?:Z2H`+Y7)><4^'>2HLX4
R*>ODB(8=^U+@;+%IT);YJ60QG'PW0\YON"4O(\0FRL5?A>>"N?;B'O3##?7
J&/AN1`\9PMX&F@?`>7Q7''@%.7T@'^OH7P7&9Z;"M><I\QGBW\9,Z<Q/UVC
GH,R7?#G;Y/4#S'_C"5[<S@'<1+\K8;G!N1K5>3PV5:L#?1#GKEC1O4>G5\C
2<@=Q'=^9F&N<'UZBN7F^_I#6Q8AACG'T)\:_MP1_EPUYR/ZCTGJ[<BG+&;S
T%>PCTPA>^V@BP9UXIS4MN*A0):S-D;==E<KE_DRR)DFU)UO#7_N+?[<5J'N
Q_2Z-TCJ)J@;S^?BOL=YK+M`T2:L6!E8A6=TDPVNDB'3YO*?8MS(_P`YC5,A
Y_!]D.W)2J\T'*HH<A+5GD+&VYTDHS@%^F"-<267PK6E,TB&)UWI/7H^%B_R
`\8?W+;@""$_;B;;<H\8OQ7Z&\+4[H#/SM-X[>[FSM/\&;BNZ,__G+UW=-BS
?W<OM)=3\#<(?\,>EE?P^GD\GRQ<ZX+Q">Z33EM14VA=X2RLL1*UDEAJ'P^%
*O`^OK<:YM`5I5:RI$0BUU\)?C\HQ2Z#.;X+QA;35E3#O:EX;XG'2KY1Q>7)
1"DZ"'7B.^*<F4IA:28I+(5KR%?!^]4@>QK\G@:?)5`6Z\$ZD.OBQ;UBJ`OK
728\7X+OL@*LX+NSEX%,*\A<QJ_#&*\BCY`BS`MXOQKN=T%9*_^MA4+9<B@T
UTVD0S!_6((RU\MDKA5TF`Y_VTE2+8PA*S#'+)ND%$Z'\GBMD_Y_A:3%:50.
:0(,JJ@O(7*]!\8V_F'Z?N0RGH_*=-N=\(>^0FQ7$W(]ED<;9)#MA#^\YP6=
\1/'1,2Q?2GO*\-D[9`Q!N`GJJ.E%L=23@N9>R/HL!%TL1)K/3Y_$)X#NU34
#]\##7U1A:X'OO>:0%W'07^L`S]E_KD>[CNHKB2N'M7X/GZZUX[RI%IL-U?2
O&VAOJZ$:\B+0#LMH(.HGP??2\W\-)*W=3^A3U`NEL?U\50Z#I5JD4>%OMH!
```

```
M,327.T;U5^B[Y+&LE9=92-NRI6D'QXPE5<%UH18+YW3@>^Y+4TD&S(>N1SY6
MOE=J*GG$4E[-<2ZM<09TC.\"?R$6)`'WNSBVN]@[!8I:*<<)?&8EXQ<N<A<>
M`+FBST0NG3Z>P?;J+G'48[\VG[__->9[9ODZ11_EW\FNKI&R=![EFB.4C_JN
M*.B#,`ZL'2M%;?!Y'-_YS^."?D3;,#Z(_;OA&?Q_"O@_5LPXV4S(H1)BJ_51
MSH92]!GHU`:X9F=`V+MSH4],Q[X<^:/8AK&=@/^+L+Z=$,M2D+T=?ON@#<+8
M-#W6_\#@YSU<[/];V%SX;MM,DKD8W[FU.6-S#O:=87SL5K(D;QIYH'/3]ASZ
M>TU.!K%*2_*6"=?,OS>&_\[S4CXV^SXDC5R/U_?BWHE_/M$"$!N9CV%0]\U9
M2J%G/?@`\.)YB_&%,E.):K,JE/O@6\UX+?OAT_,+0GDP^OV\M5-^=!3T\!QD
M^G@A9GEK<W^T$.*S7E:*2M8PGE*E6Z)<F<W)2E%C&O1)LK<,\0UCI[GP[-QU
MR"N!9[%]%*>&/_M=T`'K_R7$#WQ:J\!8VHY_-F^.;[6R#Z_UK87^"/J<"5<]
M\P+Q*)OQG,2^M-87OOD'DIL%Y;.AO0PE>\L&TV`^=Y#Y?P6A[]MPI5)?V%PO
M(CXA;SR8+"W9?R_XHV!K3IU,%N=9;+7_"'GL+2DP/R1//+D";//(.;4_@_(@
M.P.O8=ZM^S'6JV@#WR9;Y_%\A/?1O\<@GV._S?R>W31/"LS3=>\$W0<V5#4/
M9'A?P/$&SK\1EWZX7PSW!Y*]G^!U:+\M.":X`]JQ-YD<]N.S]'\B9#?%Z[_C
M8@*Y"PLN`1/+.2:6?_4PH<<_Z%&:[TH'#)P</08F_P4Q\"S(\B\78@QC79R3
M8:Q'8KR`Q1CSVP#80Y_!_\N6,.;AG%,/6_=P0>X_#+FLWB.3DS#&JJ@"']-^
M+8L42N?S"O%Z#?YO$2'7A=K(8>B'EG@>)P^<2"(5E3/)`Y8A3\<7$-LOB"4M
!/H3J_>38S"N+(%^ZG/(QTFI!U]XR2MO[;0D-6T'/[\.=6UL)>.703ZMH?F<
9&[\GC3RVT.23[JAS[UI&/__@J7)#_/83;A&E41YB\@94CM!_GX8P^RFXV^I
Z3C\KAZJ+)1A7(5]`LS?*AQ)WO8NF'O#>.?0RV!K$DFJSX>_8I`!=JBZ/9\!
WG6;=G&;J#V/*+H]2[M`_N<6N<F:^E[.!F]2;B?T11YN2PW84@HR2[@M-6"+
_KN2VW+C<.AZ_!]@H[6E%&RQ@"V5%V&+!+$K]K*]>@GZ+/S>"F6+6]@9CFF`
P>+7V7T<;R'VK)1CK10MA'ZH&-L`6Q,H<]X_N;S**J4O)%)Z/MS#\4PI?*+-
LT+X?SK8_W#R/B+E)CJOUL?/0-Y!%.U[5D6[TZ[L0T[7FO]A[UF@HZJN/??.
$`($DDR^))&9$!0:*/(+))"0FX\V0HHI1$7,TYF`-`6LQB"B1+@0:[7AN3@P
%I_++B<)VF26KX_Z0$UK5S+5]Y[6]A4SU;96GY,)44)$"<CG\LO;^YQS9^Y,
`G8)?<EZ*[/6K+OWN>>>O<\Y^^Q[[KW[0\BNY+J9-`?6>@K\O="FU0GZ92+(
(JQMVTZXSY63ZK>3@=^[[)MA+QJC--OJ";%^\!HYN\"SF+#\%/.//[8;=63G
9!*+Z^</T+_?PKDS<!_&\[?#>;0[5-]<4(]EJ<"'2XZO9WDV3*ENM+_#_%?<
ABS5?9B]8TEU'X1R)^Q9:D#'P#S',YHSKT!SYN5IPIZKU;5N7K8S`I[+H+UG
K[*]5&P/GAGY/F-BYL:+C.=,?(Z40?<M`=WU:^PCX'6B?]@G//\R?H^%XV.&
_FV\QOR4&?C!9^T;#/S<$\;/DX*?.PW\E%UC?F88^"F$\9$-_"P*XV>-X"?+
P,\,G9^KD3D#/Y&<'[<^/O@N6N<G*8P?1?`3:^`'[U$-!GE"O1.0+_BC':ZK
NS^K#<HP)]N5?1WWL[7;N)YH3?<3;5\-T5Y\F&@O;2':S[<1S96DYC6DJGF-
5C6O*4/-VS=5S7MQNIKWTHUJWL_G0KG)3!M3B=9HA7^1B6*\,/\HU=*XE+3Z
H^%X*VEMC"YJ:8R6X%\,?[FE$7VG6+Z20U^[+SG-GC]D]\,PAEM`?VP&_8'O
13#'$#PDO_<(Z)&<N3?2+X7^*$@CY=L-^N-("BFWA>F/UV$N.]E<)KD7B+F4
,H)SZ8=S9]E<)KGO$'-I?S.['LO2@(]SL#^XQY3`\F5]**?48^Z:E@OZO*6X
.P'O8/$!TMQ-4`[[AG&X5_&P?0K0GGD%VC,O3QMT/'O_9B4)O850U[/.EHWO
>;!M]U6V#?UJ#?2+Q8?@\CH9=',%TXEIF?A>#/7+')@[[#?666_H-YYG=DEP
O-?0[[I_(&]91MX`'V?@K3B,M_L$;XL-O%7\`WF;8.`-]<[ABT'>K@OC[6;!
6Y*!MRR=MVLAJV&\?2;D4Q^WIPR\G3X?RIM5\';\?)`WX>,YJ"SB.S;,P9OO
SK6O2^?G(K`?_!I\3OA$7*."OK+Y^[/TY^/+[V6YKCH".N&;Z"I\OY$N8\ZF
2'<3Z*TFT%M-H+>:0&\U@=[J,JN6)M!;7:"WFD!O[4LJ:MF7),&_&/YR2Q/H
K?=#]-9E]CW3^5X["\9_`>BL^:"S7'I^2(EDX?YG)2''D[MNI#EF=?DJ@%<U
S:)OPK@U03WV+5@=^"WX!#PWL#U2(BE'/PT)=%P^[I$28-]VEX/IN(+F]'H;
Z+A706X:F-PDN+.$W'@,]ZM&.'>:R4V"^S8A-[[?SJ\_+>Y7>/]YIHO?IQX\
'[SW/`?UKJ9M?/]4">WJ]ZXGKK(]=F]M"MY;9Y\7]WJAJT9?Y+*+-/&8;^C+
]Z\Q;8N1-JRI(Q<XS2)!VV:@77*-:9\\%Z2-NN8=07N*H"T9:'_K&M/^TSFQ
GQ']WL3T7HI[E*!M$D??N=`]C,\?E`/<PWRDX_H>YDA_UI_]['OM?H.MPG[K
?DD;=%_#XK#H^YI_8VNS`?1$(^B))M`3^T!/O`AZXJ4ZT!<__GI]T5!DI@V@
(QJL:(=MH@T8KRQ"M32`CN@$'=$PR-X&[;7/,1WQWJ#O11,NE.ZQ7<BG,,Z]
\?NW/^>\5+KGMN[^O2_(UM[Q$KZ#2'`?@+T@OA_%'&`\SD5\+WXGN*5;Y*D+
>6?Z]>WG7:;]O+#V9W_#]J^_3/N3P]J?.&C[_#MW/)Q+A?HQW0-]F,=WLY@#
SC&#G(N`LE@X)QO.V>I$KJ0ZGDL)<X+9!6Q3^3FKP*VB[B?B:'Q?HC!_Z#GT
=]WXCI@<(L1&MP+\JVZTKUG["COOFL.^@7JZV3OMS%]V<Y_#P6(LL_J^.?35
[O"8R:'O:%@]VUSZ,M3C/,RECJV*UJSC[8`_JFC[=-PVCSHVP9K4<17P!Q7M
>1WW`?Y#17M6QY4LZEBG:$X==P&^5M%VZ3B93QV5BK8S0!_PNQ7MR0!]P%<I
VN,!^@NHXW9%4P/T`5^N:+4!^H`O@[U[@'XV=2Q1M(T!^H#?K&@/!.CG4$>A
HFT(T`=\L:)5!>@#OE#1U@3H+Z2.^8IF#]`'?(ZB503H`SY3T58&Z"^BCDQ%
*P_0!_P&12L+T,^ECLF*5AJ@#_@D12L)T`<\1=&*`_3SJ"-1T90`_3S,OZSE
!N@#/D'1L@/T%U/,13XO0!_P"$6;U1W^_?PW0;DA^70:G'_S</_>3EU&E'R:
`65?0!GF2^9MY8,\FC-WU*[RUJVYT_NXG5A2H(Y77&=G,:X(38"RW;PLDY41
B>+:,W4;ZMDE.@[PUXW7MDMLS3UCO-8F4PG*?F*LI\KT`N"]QC*?3,\`?M)8
IICH"<#WA/!GHL<`7Q'"GYD>.<SL#N`:,^W&<R'TS-0'^$>'T1\&]'4=H1^?
XGXL/\!O^74L)CS_=BX)/YC5A-9^!>>B%>UUK+,Z6`??<[,ZH,\CT"</YD7Z
LG_O:F@?XP"<.0:\A/0-GH.A?MMA$0</Z/]&T+_E"O0K!/UGKT#_TSY.O_,+
U'$#8C_`/(VB#4#WXR[#>+E&T>>A['^ZC#R.HGNA[!-CF1)!40Y\QC)7!/UG
*-.,[9'1]$DHN\4X)\IH6@=X2<@\C*9;`3\60G<T?03*"D+F/9)N!/SM$+J1
M]'XHFVZL1\;0=8<'R94"NG+686:GF'D)VJA9>Z=7BB24Q,#])H)4$[+]N5-=
M`W4Q^X90"WOF*%)>!.ME]?Y9U%'BR%X->V?'DWSO3&I9GE@-X]62S1Q>A?!&
M#M^!\`,<+D=X`X?1IIA4<7@9PFLXO!1A.X=QWT<J.(QV=F0EAPL1+N=P/L)E
M',Y%N)3#:']'2C@\'^%B#F/L1Z)P&&/MD5P.HYT4R>9P)L+S.#P5X5D<1CL5
M,H/#Z0A/XS#&VB-3.,SB&-HXC/Y?)(W#"0@G<QCMHD@\A_'[.HGA,'YG)5$<
M9K$((CG,_+_,'.9^X1S6OY4'YBE2S-.LV0/F2:XEVF*X5M[,QTG>2+2%>'R`
M:-EXW,#'2:XB&MJ/R6N(AK%'93L?)[F":&@_*J\D&K[CD\OY.,EE1)N&QU*B
MH6^+7,+'22[&/.)P5%A.]'@YEX^3G(UQT^`XCVAHBR?/XN,DSR`:QDJ2IQ$-
MX_S(4_@XR3:B,5N\-.Z3+"?S<9+C>;YM.89HN#^6H_@XR9'\NX]LQAS?:,,7
M'*<!MGBP7STCOD,[;KC).Q_&L69=HJ7F4:L%;=$5D0<1OR'R>\@8NKN+VY=O
MAK:_@CV,2GZTG$2X'MPFD_7M$6A'$L3?!MT%:RN`V\V(UP5PURC`536`IX\A
MY4EL?K<O5T09VK\;\;?Q'8&!A@_.+\)KU"<"=5SB&E7'@0\61W8J[%LLQ%E"
M@CYZ)^*)LUC@&*/@"\`5@;\(=?,$O`3@A0*.A3%8(&`[E*.\N)8JK2WP_/$I
M_%&OH"^#\W/V/`)ZB;`<$OB\DW:&Y6/(\L&>BD)=W^^4UFH\/LKY^6^@[ZOD
MO+P+<`>VIQ#Z!SBBO!P$F@@?UN_KZABT'\STFQ(SNV`_G=FE[RO&LF>N2DN!
M=_7\)=XUA<N\]R[[GG?M[>7>=*CSN9_K9K[G&$?3Q+RBS+QRB=N'A?JQ_\>@
M^\\/H1W)KNS>3L@"W,=C?$RT*>'R$D4CN%ZU@)[<,Z@M!+1SSL]MR#G?450#
MW'&N8K>OSO$*K)UZ>(SK0-NM'IA[OJ>*XOG__*QM]DV7TQO/RGN@'+\W7XZ>
MVTC/-9Y^#/A'W,<\BX_'>/H7/\9_2J3W^\/B@@RRY\+U]`[46S:!./$[^#.\
M_4R7P5>IZ0I^=LPF'/8)QWJ!;Q,Y6%-)+-LFR=4>F?F0["('"C3K%-)7(HO^
MVR8PFU'<0_S"S^PU=]F@#GZKLQV0,/[0+HR7BV/@D-AS,/,]0IS%TS#DI=?/
MX3B&VA2_,>!]%8[9W7Z6.SYSAI_'L:CW\Q@,,_T\5\A#\.QQGY_MO0[>"T?4
MUZO%FI#.T@YII>N$1*3ZN8<?]7X%<KM1)MKF"*(]/)9HFR80[2$+T4YTPO6)
M1`N-2?'6P&=,X$N/FWK:)*//@)C3";3*+]8>CH,8`YQKG`/)1F@&S,%D@[\8
MG_<)++_KG7Y\1HWL:P>]@<<-!/,&D0.HGU^08_I45I[,SL<&9"^:+O7SF&Q<
MCJ/I=_P\'RJ7V6A:Y.<^WGR/'D/S_=Q'A%\?0Q?Y,1Z+S-K%(](]*>AND\V,
M[C8Y(I2N&D._;:3;'D._9:1+8NGU1KI*+$TWTE5CZ76`'[_0OW<3^]:4Z/X^
MP!@/@,G1&UR>C/*"XV@UXS?1!'<!M)'>'DLOG&)S#JJ0+L_XF#T#QP_FIZ#G
M%#\)\_5N)Y<EU!MK8/UMW[K*:RLAW[.6DNJ25%(.^YQ#A?MG4V5EZ#<9.\B8
M/&/#;NF!_1WW@%Y`&P]<]S>/)4ZTU_BCD'&K:FI%_%WV_AGC@J)-69R;[1&(
MA=[7KS]C6>A'G4%Y^;KU>A_41;\H"=;KND[#,Y9JH>A7_$ZG\)<JDNA_G>!^
MLRN@KJNH('1??R/W_5[;QWV^&K%.1D'HOOY&[A=W_`ONL]7;&^33;EB_'X/.
MQKZR&)RQ:!\0WM\X.KN?SV](N1)'IPL;NU^+\?-%P^Z/]2>._A3ZXJQ4O&HB
M\)1&JAUP#W\:RN+&$^=9[F_[=^JX9):?Z<!1KB]`EOF::X^C:$>/X[;%,`?M
MXX)]N\N@LXQ]=EHF65Z8(U5[8"Q_`-=^$46<^7#<Z>O?6]3)]/D"/K_Q=`W@
MF.ODL?-\G&[6VU2+6JV1I/KWT,;43L,]48VG=^C\@+Q+R4IKNIG$2LFD=8-H
M`[^Y'/5Q6=@*_?K,-U`6;C;(0K&0A4-'+R\+7<>Y+%BO(`OW"5FHZAUD/DD"
M?>T2G\^$2WRLT==@X+PGT'^]I.\5$FA:)^O3[G`;H<%B5>-]H(O;:_+Y?Z-`
M<V$^2+UMT`<_A;;SB=SG@WL6'E&7=4(=*T'_$W.?BY5'L/,I_/N0F_FC8`P@
MD]AOMB?0TS[=?UOM0+\ZWL=$NE7G74FDQWP\CBCO2R+M`7QL0!\FTFX?]Z/E
M^C")<I]%D_L3GWCF,]I9COQ&?B._D=__KU^\U69Z?ZC^(_1'Z(_0'Z$_0G^$
M_@C]__L_/%\.*?TC0TS_RR&F?WJ(Z5\<8OIF>6CICQMB^I8AIC]QB.G;AIC^
MU"&F/W.(Z6<-,?W<(:9?-,3TE\AHX][6TC6NK:4[KJWEL]2VEIZ,MI;>Z6TM
MQ^:VM7RYJ*VEKZBMY>32MI93*]I:SMS5UJ*M;FLYO[ZMY6)-6TO_EK86?$_)
MWX$2^K-S_7O[MWB:+]9XFL^O]S1KJSW-9^[R-)]:X6D^N=33W%?D:?YRD:?Y
MV%Q/<^]T3W-/AJ?YLU1/<W><I[EKG*>YT^1IOB??]+[/]`*^[[PVO,5PWC8-
M1]YLG+=;AR-OLSAOTX8;;TI0WJ3AR)N0MP^U8<B;D+=_'XZ\"7G;.=QX4X/R
M=N]PY$W(V^+AR)N0MZ3AR)N0M^-GAQEO[4%Y>V<X\B;DK7$X\B;D;<MPY$W(
M6_DUYFTH]X\C_Y'_R'_D/_(?^7^3?VC<.<D=5UKJ5=%/MJJ8V>TIS!?G.AZ?
M%?9$&"<P1B;.`\(N:]Y$XHR3R`%VC:LLQT5(EAZCN"V9.-WLWAWGWO9#N:\J
MEI3#L7I:++-5WD7:DRA9Z=CL("3&)NPW]T`[-971U+Y,HET6XK07%N2HK,UX
M]XM)/$>'@OGSF-TUY^=UX(?1MT=34BK1O\%U.[$>T/\)'-FY?64Y=HEDH>T\
MXE*9N97S<!TUMO57B3@?$S0<AO(7@,;#`]OJ8&T5RQIKBTP*:>L_H:UUHJT"
M0_E.:&L-X;'1LJ.(TR'@L>.(\VX!3QY#G&C;J`K;_S]&$N<=`F?V_X"O$'4O
M`EPFX!_#==\5\"\!7B+@>V$LOH/7KU,T85^YBRBA_#X'_!*3NCQ]@MJQ`^C4
M;5(T!_"Z9QWZ\<>['THD3@5D@L@L9RT?/W42W0[GT7;0)*L=3T$;+%]4/*^3
MR6WK8)Q#:6V#>AF"M[,@0S8!1Z00YW4$<T!*[UEK25\SBVG)Y:DSEC@3V3GY
M/6LNZ8O$G-@L)S#(*HO'P.O='4^<:.-L9_%=@G([4V;^&S!/UA!YK@!>KA2#
M$OU<'OZ@?^]-<#W:-VX$>#O033<7:NA[\";P>+]>%BG*=-Y`)N.3%2TN3='6
M0ITN;@,*8V^E/"YYG#L7QYVHEAK@^9\^8';/[GIC7358=S;4/2CJ5$00)X[9
MKT;S8PW@E;+2JI@QCFG;@QC35)7UM685MHUR)O;9"NW$P;$<SCEE&^4QJ^/<
MC8_6G,7S29+PN=Y*^M+[6>Q2"_J?N4`N3#+W9>#]L#'>D"Y>-T82:Q'62#NL
M6Q?($.8^GG*"^V,L%//\;<'[*.`=[9(QEA_OJXWJ>2.:X-I3!/DDAVIV)-/K
M=1D)OZ8]])HC^C4=\VFRN&9T^#4D/>2:O_%KCM?L&!.(QVV-$/*BA-8]1/A\
MC1;U$EA?$MQV.?^#HWV\G^'^Z;*9Q>[+*83V"EUI-.-"K;>H:G6VR4QR,C[?
MS.%(@#_:R.$H@'__`(=C`'YC`X?C`7ZYBL/)`#^_AL-I`.^T<]@&<&T%AZ<`
MO&$EAZ<!7%'.X1D`EY9Q>!;`N:4<G@?PC!(.9P.<5LSA7)*3'J7L+JAR9$L*
MR;$6DQR6^Y:0`[B.[JFR9?-8=1*+-5=@5BT2]-<1`4?HX_9(.$+_&L;",8I?
MJ_N>$O.K'07$U7&L?Z!?/HE\%9[%G^[X-.3<P%RN6@9?IV?A'G;&)+M/FTSN
```

```
MDR:S^RO3*/=;X\B!?P&9S"FR@)Q/="\&/5(Y1\FQSU=R"FW$;5\(1X6X'8OA
M:)?<!682ZYA#<FRY)5[[?#A6E'GM"Z'_M2N]CL4DQ[EU]`HE\D\I\4_&IV)[
M6;&8^UVF.[8^XHTC<GT=YD:'=>]P*JU2U>1LUDZ$:&>L:&<"R=GV-,G&_&-8
M;[.PLT=[<3E:77[KT?Z];&WN2*>Z7L/R6]`^''.6F9+=Z\.N*3RJYY/@=1<'
MZB:Y[6%U%X35G1NHF^B^+:SNC+"ZTP)X`L.G"#R=^7M,9==8CXH8#7I<C0QC
M[BP^MV:8]YM@WG<.,N]FF/=1,.\_NHIYGQ(5.N_M,==VWE^+N?IY/W<I=*S?
MZQE\WM_M"<[[L;!KWNH)G9_VGN"\^\/JMH;5/=@3G/<_A]7]15A==T_HO+_4
M,W#>&T29!'!7$K^O_CTR\-1@:Q_F'];_"3U/46D/ZE9>QO0(Z!<=!WV#.95S
MK)'!,BOH&2OH3LQ++8'>M(*^M(*>M()^M()>M((^Y/M!7M\T]^D.Z2V2C;X<
MZ`,@8QM$[L68[CIOF&_;^CBIG@;W582ESVN]MDCRK@PT9<:K7$^J*E\G%25>
MC(MM`YX(R`'4R<D'?O*!GWS@)3^9ZT"]O72\9^_:GH(XVT?8S=3V%_O90?WM
M4@GL)\B*,P;?-HP9\E<8PVU/D&H;M%6S6TVIF416.*$M9Z69JA_6G&7R`/.,
MOJ`NJ+L'ROZ7O:^!DZ*Z\KW=/<`(J,A\,`)*SQBCCL94=WTK2LV`!I48@YB8
M9`U=55T5_(A104.(B0WJA@1CZ)&(2\RCT1C-N)MEC1@G64,+[CZ-218A&DW<
MV*#1Z*#R)4PC,N^<>T]5=4_/D.R^]WOY_=[KFE_/O^K6_3SWW'//N??6O5V_
M%=^EE"KTH;9!_HV7T),7-/2QTIUS-_Q'17N>4GD^3W3.1(%)CS6MLGKPF\R%
M\=@*_(X,RR)+,\M-K.E9_([QY__)O\W8\)-)7`_E-D.&"=WMVE;L1YMZ*\^3
MV'ZK_0ACL14L^]E'\)G3!N)#-WS&..)MW>5E<6&#)*9VEUVZ9]:L/OPNZ32L
M*RC7;>!VR^+DP'SXX?>02Z&MMB^P^MIOGKEM!F/]I^(>J?3-*/;AE?O`)VE/
M\]BA.=N2K/NY'1=9CR!O/,//97>>2[+,<]O!#?L_MG+M[T>J-Y15!^C[F*9[
M+7[F4?^@T/4JZ8'WS5`6!6RK6QK8OY10#P9:X%Z[2-]8K+@']<RC`GVF<%H^
M.%_IH<-"!T$]^J9#YV];V&SU_`[B60T!#W0<Q_>`^]JQ;)4*-/E)Q[8UN24W
M#4#]]-ZU1!O`=TN/Y?I,)YY[4QGFNIHP\K>",(LHS!%YA.@0U&D+T!+K4X-R
M-AV:N^TQR"/G5YY><^\\B'-#J]53!)VTZ4//KUD89Q/1_3,\K:;.8*_Y(,T1
MOT^MH'M`\SLK:-[<]E^@<RFB\\7#TMG0.,TF"IJ==`0ZGS*4SA2F]0ATGGQ$
M.M>>IU,X!<_,PG/3XM!')CK1?LF"[1KP&G[7G.3?#7?FE]YLW17(@$E0MH\+
M>W<EU^5CG"=7LCFS^C+!_25\C$"$SU2''P/A\?MX/(]K=3S^K'6&U9,$&G._
M_)RH>"AO#GT@Z(C[593.M'I8X*_4R?NP5X'>>\D/[L-0B'^E!_?"ORF6FYN$
M7^N$NR<7XR>FIR6F78WW)8B;9%WG_2?%\F?_:7`UV"(;]O-OQG-S<1^I90UL
MWE&\76/93\_CV6`[>%JMG2]26E-HC`/[.]2U,O%6+H<8\.W7WL8SXO%L^*;^
M/Q\6<N/8(+Y,$%\SE[5/07S+@%[=(&\R>/Z.U65VS^HRL>S?!#K]='!P4O5W
M=?\V;+L)SD+`\[KV05VZHAY75M5CX?2\75$/#T#:<Z@<'0W"!MI^%)LGZ'MZ
ME1V\!OSFAMC,-N1O>L`'8._>=AB_0V;]KS)NWZYD8._B7N#MG*ZL_]9&<E\.
M=O`$MAML,9%6\@RRH5IZ-W_=NNN6#[B]O#()_OBWZ<OC91YN7Q?8STW]I5%L
MWDE0MY<Q0W<!'^/GM[!F=,-]T%UV8EK431,_D^NFU_BY21N"MAO44]/;D<YR
MD/*)]5J@<@;Y.07>H9[1'-9AE-];X?VED-^]4$^@)_2`&-B#:>P8_,[6'<<6
M]F"<V!?\",KZ&/C!_-UX<(;V/-^?/=99*Y^B_09>IV_4WP">31[ZW#:6_%K/
MICL6]3P`]8GT:8>ZG0%U!FU'T+&`^1K-QP)>38SI_0CQ*M_?98$S$&^P>N:/
M9[N[<)\+B,<Y/'=;-\2)[V+P#G20W79,O/,/7[#MW^'=J`7^P#KT"VVO`_27
M^6Q,?S(AOB,7?%*=Y@1*<UK8?CY2];Z1WD\*:2G.=\)W,7J'WZ0_L&]QSP,L
MT3MJP8<';@,[\H&!"WKP'NOY@8:+'L%[T983O2OPV\_#<[9M/AK'BH;NWU`]
M7G,7T(V/(9YHE;$MXUA%]DRN;ZR,@^Z`>V[A]Z9)D$D%D`N<[PYUX7XY?"]>
M0>>/Y'F;!7=L!]L.#:X^+2;:00)X_E4\UZLR/(-V`.%/"L*7L,R)7AX'O.-G
ML$`<;3&^3TXDXY)G5HV+]8&?8\'/,J@'2_C;,)_&F'@ZH--T,[8"SPFZ;QJE
ME:F.XR&(XW#09B&OJS\0;=8!_V"CK$0;193Q3%XO3IM5=J=:Y36'^'F/<X^C
M<K+QW>7E'^!^!XSS=N!V*]$TH#/NB;8V'NM?U;-T309H^]4/Z!Q0N%^,[3S.
MGL7QD0YHNSM?&EP=]@.EZGPOH?1W!WF'?#:=895Q;R"?RG!2`^4#WL5`=\/W
MJ&OB7D)7D)]X(O+3)(GPGPK"QX>$EZ+P'P_"!W68_&A5_N9"_DZNR%L,THXO
MP+V4P78$?28^@?N;BWODGW!8[!DBZJ<Z'MR+^P>5\4@CQS.N,IY"=3QG0CP]
M0L=>V8YU"OE!?789ZJQQL$V/8TKX3HK>O1_CXYB=I8E,N?DX<9^#^[#=0+V-
MI3K&-H1[GPH;H9GW"R=!'*]!_X/[$(.^)/)6JL[;&#SC&<=NVA[;BN4)POSN
M`['_27%\H'=(5>'*[P^NOIYHT[RJF[?9IGNZRR\\*_0U04\I'`]]&_QGR7_L
M'N&?0;AV-BO_YU?%V;V"=M7IO`+A/@7AL,R@<ZY<MJ:[+,HAY>V[N\O=$![]
M/0_^EL*[S-TB3MP+]F&@7VD,4\,PR515F*??Q^_;*^K%`MK/@CJ&L/<2[2VH
MTT`G".+H`G_=X`_C^"G$@?JMS/O5:5?O';-V\NGPO.I9L3]-T!\7#@F>Q;/+
M&J"_P?;33N4*ZG(U^'&@'E'N`7\]BSRPA/)1C#-E%(5K(CHV$=T9T!-T[A5?
MAC2Q3[;W#L[[(?@=6JX[7Q7MMGAP<,-FBBO<@WFX<YEHGX!0MV1"CVFZ(U[&
MOGKN%O'-/N>!.[K+J&/A^4QW;R$>2`1]=W.O>4CL(X.\.6N+L%&'.U-IV/16
MBO2F5Z:W,DIO":5G5:1W0D5Z9_Z%],9MX72[NA3('.C+Q5Q64V=L"]]OE-N5
M+%.X@;&FL:B;+17S'9UEL+=1/SWX'V(?E$]37HKQ,"^=!]\?(7W:?RE(L\3W
M7KEG*YM^W4]`;_E6L&<2I=7[&J2!8]GG;N'[=?>6*`WDP^WO#]U+,BKC#CK3
M*[")7^4V<8+;Q">!35R"9^P3'^(Z8.:Y:6S6[MEP#[;6!M&.4U5VP,&#0C?`
M,?NF?5_?%K3EV(*%!NI`!7[F"/#W5';IS%FW;?T9Y(VW>VSOX`?Z&1/O9=:T
MPAYKF9EC+#.9!-DV5HS!Y(YA_*RA2IIMYV>9_;7VO!B7PC!Q*+,UI,PS*\J\
MG.]')<K\9J`#%U+YH*P;#XJ]L'A;@CPG&]F*!\>Q54/'P/[;:96BM'XP3%H]
M%6EQ.M\FYGLY/0O-^?B^#5OCJ^)E=DQA:[)![,47OAN@=V/%.]P_Z!8:4^F"
M]^T-5GG:^"Y^[L]H"I>`='$,!F7>>R`C1IIG4Z:S2U%OCT-Y`_JC[A&'\N+]
M<GZ?@?M9NT\+^^DTWYLI*.\"XB.T/R\=AH^*%7R$^OU)'<NVGD*\Q'@^FU;$
M&RTS-MXRNY*QSH[Q[(9B`^Y)TGSUJR"'[P<ZW78,FXAG+]X&86X#F=*`M$/9
```

```
M"_E?=BZ[%.T^W,<[L",W@YR=GX3V#;)@%;Q/TI[`Z#^P$\\Z&.6AB7C8`1[F
M\2;CO8&_0U0^K-/F1F8N/7?V-IQ3"7B[6N;6CH6@;/@TI(MG`B,O:4#O9KYG
M30[/@G\4Y<YY@X'=-*GS;K'7']^+"<\]GDCVY/0*/]\F/R3?>B\9%/M*X?W'
M!X/]N-%>;NUO!WM9Y7W$2.,80L;@OJ:OD5V&LA'W<BKHUE?8>'%N*IX)AO;5
M4-F$X5X8*5SCD<,]/5*X\4<.][.*<'@FP3H(U]Y\WK:1_/]X./]MYX_H_[[A
M_-\\YM*1_-\]4CG:CER.Y2.%FWKD<$M&"M=\Y'#7C!0N.5RX)ZKZ-QQ/_CL(
M'YRAF\2]M6D?[?L^;7WE_OLM$]>+CQ%VZ!;<,[MR;./SY:%G%]?N:[=/C!MV
MXIA+>X(IF1C;U;*S"W2$%MY_XC/7*:!]+X1[U+F^/-HJ7_V;4%_=@GMPB_V5
M)O5V09K%&/:OS;V;OF[=Y9[?9392>\9]7WE\NS!^UF_QLX:;>YW#XFSXLZ`=
MO3*:7;\7Z!+P`;:CNT:+]3%I?@9E:R=O6Z.8T@0R:1!T1WR'\POM$[X[^?[#
M?(]))9B[1MK_#W##,%8#4W!/5TR#O\-UQB#OT!:YXGW1OI/"%MF">XD'94!:
MCBOS/2+Y>7NXK@#/.\XM61B..<1HS`'/\P`Z=.(XVS;00?&LO0T3[YDL]G=O
MW&V#;,"T+;'/9.=XTI'&5N:+1?G2*5^90X/J[VO&T88??Z:S2#OQC(+]0$M.
M<QIWPCJ]"?2E>W\]N+J08/J-=R[JX>\'NLH;H4Z0YFLA'U#?HNYQO![J'MW6
M!?RP4_@M\+/86G#^K+,#\KI@,+3E@$\%3W00+_Y\0/#IW307<ONO`SVSJ7?Y
MKZ.YE^'&D+`\N+\PUN6_X3CE36*-T*;!$72W(6$>KPBSH29,;?^QC\(N+0M;
MI(59Y540_O$/1'^PDMHD[D5XXYDSMY4JQMIP+'=.&??@Q/8HYRO=EP`-<+U2
M8;35`_*6GT-?^,VB'AOB@#8[%^O[?.*9`U#_J)/HF&>P\8H5:2`]Y2"-I)*O
M=+<'^%YF<^^&-%XYANU>%Q,Z\+</B[-J<I0>VJNG!VE-7#MY[9V+]FR'.L=Q
MJU4-M>E-"M++*.%:$70_#]+;#_D['L)ANJ+NE:IR3\?UZ,0_P7SKZT>S59'<
MJO8O#0@Z?_D#L7>9=GCPT;>`]S&/2Z']O%[9#H;I5['^L?Z.@SPW_4KP=QQX
M]I9XO+\0\/`N?&;]_'SPF"BK"^XNN+_,:4BR"FT:X/^7?\7/O_TE^U<^)O;H
M'RN><0S+`;\SP68,XNJ&]%S0$Y\9"/=PFXMUO/;P4/ZCN6#([Y_`;ZPBOS-8
M=7YQ_`[WPL/U:4%>_V5`Z"%!7@N\?<5X>M@F;ST<S3NC+@8ZW:YEI\XLS[KG
M\]N0W]8FD_VXAR"^ZUYTV39,<V:,TOS0S#+C;JS_OB`?<V;Q,UA?Y7U7HO-(
M^Z7BV4JHA]T$OWV)AL[W$J/Q;*YRLH'X'^3_#T%>@(ZI"%Y6\SM`=]P,OW\^
M$/";RGEC&<CA^^'W(W#?=S26<5+G=I"QSPY26QS'YAG@_B2$Y?EL`#WWT,?R
M[-",?%"&D\93&1IKW\V&\!M'B_53%P=\B><]Q*URD*=OO=H&T.QC(=]6OU\&
M[V?S]7M66?0%DVAOXN;.UQ)MG7\">7\3M;D2M&_<A_$5R/L.H.5:<-L.;F`#
M7)4<Q>9!F79O!YW\3R0OMT/\F\[%\[ES6V=A_P/MY\713'D!W&PH4P>4MPO*
M\DH\UI\8S>8A?V2AG+=RMT3_`H@3$?>PO/T2J^?V!K8!]^SNAGNHVPWXK@O>
MW0;/#0U\3`GJ0Q/EXW/J3;T7'HCL5^RS<*TBTFS=@<%YCT*;;`O:?T:$P_R^
M"K_I$.[#\.YGX&<T#Y^;BW-"F%>4":_0??NAP7DOC]C'U>ZG&IPA]3[PFC$=
MUPO$08])=%Y,9U,J'U3K$WA^N#P6=8DI-*\@=`EK'%-DX,&?`JV[N-W4UHGM
M`>LR=M+QE[:S:6G<@QYE&+8_B[?Q*6'?MN5`E-XTLN/Q>3H]+QT3A=E!^L1F
M"*.%?3ZX4W]_=EGT]U$_JN4Q7*"'O+E_<#7NQ1Z;4<'GT):/%_.+?!P=95$W
MU'TWN&-[+G]C[61\=U!?.WD?Q"7&KW`]Y/'`GVW`GTW`GY.`/R=WO@[\.IYL
M'3QS;"'4WZH.KKM<C[H+]DE(LZ-YWSVE]V.'A&Z%NE2A/*B.(7>4/U;P#O)=
M'!CD=(\?$GK/\&<2!?MMUL[S!GWRO0>B/GDI],GK3F%E<89@4^_CAP2]6;Q"
M;VTDV5+2JG3CU?M)+D_@8Z:\[T6YGVL4[>VE0WS_4&X3_N(0W]^YLTVL&5&V
MGUM-^X#NH6PAMP&@NP)A+HN)_4&W`ZU1-W2A_WT-Z#R*K\7(S5U_6(R!X=ZG
M#.S3RKBG)?7^PDE\7515V#]!W-BFUE'8^W$_]@8Q[W++'8OV/,G(7DCJ8=_Z
M62CSVHI^&.=P&\97],.9R"_&=S'1Z+OO5Z]UJ]D/G<8#9T.;<>&W"GX_9,*6
MB$-_@;+B/I!5T/Z4DT.^%FG%SQ%AY/UBWW(<`T;9$?01.,_G3!?TC@%-DB#/
MY@-ML0SME^"X'/!]4NE_*"'H'VL<V<\WXZ+<,4CS5JC#+*YA'\=6'4/R;*D8
M2U^/Y_#LN-FZ"V7>T9`O"_I;S),8]VCJ_2WEM7IL<@3YM'#4CPZ1?/HLEU&)
SC+O"T=U[H/^\"0Z7Q#/,]PX1%;]A(\U3.V=@',-K"6P3=:SI)''LS/O`YGP
*OQ>>P]E26XNKI?'=A?81;@N+(/KQB8LG9QDD],S$M.NQGN,\S.0+K:-1^'W
/,8;$^UT![S#]O$]7D9<WRW2PG:#:3WS'K=GM^+:(1RWQ_;3/F'MFHU<CDT-
Y=A1`Z+-L"#/A2C/?/P?XCDEDF-;D-_CP.]3N!R+]]\?XW)L"\JQF;P])?IQ
/7P7^`_Z;)P3"'26][^!LFUJA6R;0K)M,LBV)I!MDT"V'=_Y!LB>N8,B;T/E
6]=A<6:SQN.8VOL[?EY;ZPAG+?Y[K>W5%-D.-^ROMAU^=%"TI>^_+V34DPV<
7[>P6=WE4H)MX/<@KW)@@],</?"A$>K:."YVU7LXMQB-U:-;"M(Y+5S'T=P[
`?CY9'I^$-)\!_I3'C>N:QD<5(^C=X6#8KZ_\ETP[O4/F%<^M]_4>S?<OTGW
=\']["^G^=BI'U^#@O/WDM@+>_Y'NOU;Q_IV@3Q]AK@2_,RG%V:.@QVSIN+>[
'&O<M*8C%_\>NG?$V0:DXV>(?K=`O#B6%_)5TB0:-7$=I1UHI&!\</^%@SBO
U]R+^O"1YA$X#0K=93PK+#F>/9L`FKZ(^Z.#/'#`#=\G"GPN;#V>L85C^MWC
$\^.?D^,.V?`SWZXW\[PNVJ(<X3Y!$SK%"K'Y][G<XO*-L;E_9;F'W27,V-_
NZ;Y5\W?:XZ+_F(N^(F!'.+IP_OLV,UK1%LR\UC>6W^5^-Z?]HGXSC@HRAW8
U*<>%&,J1_KN@\8)^-JI&.4+QXE+(#_FQ42^8E`?N;%/KFG>$OO>HXSX])+S
^O!LT@+U(T<?Y/*0RP1\/NI@M+?TD<[(JTQ_>YGV[1Z:_@^&27]>=?IOEZO3
?[/\7T__24H_,33]?QPF_4QU^K\>DOXS(Z8__%@UR@O_/3$??2/(BIMNL,K?
+XOZVPRTM!)B_.LW6$=Q<?\2CJ7'J0V43'[.G0-A70B;V2?"/GHP&-=N[L7Y
F#CP4@[ZLG4X+@?^.MX3_C"MS]%XV1JXOZE![)G>]9[80WT)S\LD+C-$7B;U
/ASF95+O8Y5YP3/F*O)RSCX1-G\P..]@4N^]%7E92GD9)'^8EL'S,JGWRT0#
C`]U$AQS+P1]=*8ZG9/V81Z@GZYP>VN?6,]0&<?\RC@*U7%,'":.E_:AO*F.
X]K*.$K5<<2&B>/I?6*L"\>@[KXJ.3#<V,6,PK'7;U\-?3]@"?HRB'OLD<:G
<$T5Y\7L>7WJXLSCN269Q[=#W_8.K6TN0CWQ]PO.Z[MI2?"^M?/UX'U<O(]=
<U[?ZB7VXVA/VX`@OU?G0"ZAGXWQ25<O%V,!6V+7G=>WBOMKYOY^BW8WV:RO
```

```
MQ)NO+D'>VR'/E>O!CKAVD\K!\UAJSL>G6WW+ECB/X_E9[8LS`UQG`QNX$6S9
M^"BFJ$#K#1UGY\%F6O]"]]EY'?#`LK/S"R&\O:F[7$C$\DN/B:W`<X:>>!G'
M-F-\_+'E/SG/*N*YM?<X/)\C?)[4>S0\XS>=/!^Y67V\SP(]$>-=N*,YG^FT
MRD"[@:_O$^USX;E67]/BA8]?>FA015NXD=;HCB8=LH$P6%,]_#E"8LSGVJ<A
M+T_=T/,%B!O[OO:G0>=?8`\,G2/E[WX%[Q;;`[@>:P:&N^V&GNOWB?[GTT'X
M+<)/%+YZ7/@]DG>XICXHGWVFU>>DK3[W+*M/M(FS\TNA?K$N;EWB/O[QO1#W
M4#^EL_/KP,]]X.=^\#,+_!3XG'H+Z26MO>/V#3>F*\I3!-[I#/+\6Y'G#^^K
M_#ZB=CSI)X/BS)VI^\)S';8P"+L/Z@;;9NL^H6\5Q=P[R*'IXON01!O7"Z;L
M%>T2OR\1ZV!!9]@KY+0AQL^X#I"!^!+[Q/JN9#S2RXIBC&H]GG%9&>^HJGB;
MN$[V*XJW(XCWQ>[R.HAW!FOFXW?<[661UC36UH_K)P3MI^<%_9IZW]T#<AC\
MO0!U]&2W)72/TG20-?:`2'M2[^O@Y\ESK3+2?AUW%SKA/^Z-SFU*!OW0D>8,
MFJ)^Z*F]43^$Z^I8(.>2YXCO($&N81IXMDM#4+Y2=_F[2TBNC:#C89RY05K;
M`W'T4CHXEGH7G9_QU<KT,N=4V>L/[Q%CLGRN!]ZW%\[)G_@R7Q.KA&ZE<_+'
MO\S70D=NF7/SS2^+>0DNQ_ZUN^P`K7B^F[OPN^;>S)+V@6H]J9;_*K_!VIM(
M]&;&6W@VD."UTKE5:Z]R>P1/5+I]#,H+-EWS-7O%65&+(/U54+>YT1`/\(&_
M5_`</I?@V2'Z8#_\I0_$F5*7`9T>#?E[!C\K-?C>Q:4TA9L81^^$./`\OE6C
MK;X"_'A[`UZVXTS!\Z=P?&/M7+;[39#Q@Q>SW0_%B`\S,XC'Q1C_G#T!CT=N
M31`WC@$%Y0C.6%EVHM5WQ1^@;4,^L"RQKR8'\,S6'/#IDTLJ^KXJ7GQ\V._>
MD!^3>VEM'J-R%V:$ZSX^#/FZ#N=39L;R\='07T"ZNW$=[='LT0?I_)<)1-<,
M?@\`\@)I/8[J`,^C:0SX\*-6&=LR\N<JU,GA][M#(3^N9*495?,>8R#M3W)^
MQ/FOUM[9>"9+7/`=/D\\C/.@?`QD?0+Z6%%GC+>A94O<@<WPV[];Z#?H'\<2
M5]%:],40INW5YNC[N"E67R.4K>]LJ\PNLLKK9`MLE>3`2T(?78^R9-T,X78B
MY>GZPUR^0IMHSG?,`IUP2<<`KF?[,OB9,8[MQG4:!0B#X\RGB3Y^?47;6,_:
MHK9QE&CGZQF\9[N^MHW'"^^=JS(#7:S]6S;H-,'WU<B/]T-]?'20MTT>QD;9
MU\AV_TB,3:[OGMI5;F?Q7GL(/XCOYYX>X1OU6&\3\$,F^GX"^)15R8C[@)ZG
MDP[>LT?4Z[/[!8TKQ_E0[EOAN`2KJM<\U<F+$"[XMAK:\43\/B_[!SY&LSL)
M]/HN\,?&L6PWQ#-O//E[$N,_$*P+`7K`\^!^(2?Y.7>X-I7R4]P_W!G?D;R,
M(>^?8?646*!GLI#O%T`>31YGK/?R/8)G.:T;NLOS]HCUDO=3G>)XY"?V5*W[
M!3Z,5<FK>;MQ?:.@VRW#C(O>O3<ZJZP'WC<'M,M4Q]--\0BZ5K\[>W=TEM&6
M/8(WFOX<+R\\QLK;FV;D.?^U)<H+=XC[IJF)\DAK#YN!-KE*/BC%>+L*TDI"
M6A\EFD\@^EQ`Y?H2SF'$Q1J>Y0>X'L_OOWR@@B>2\7#]+<8WGO)^*<1Q/(O.
M:^H@^EQ<R4^9ZK`?[!)A9^T7YS)%<X'+YF[Y?30W>"/E[ZS](Z]Y1)V<QDG6
M=_QCU3C)W&"<Y%B*YY,'^#ITXO%XU1C)B[N$#1#8JR?L%VM4N.Y>,7;!F/@F
M<.@8!LKEG^T6[0O'24A_7L]V0EL'O6'IDN3C1QIW$'+IO#Z0"=??,8;-PSCQ
MFU3.`Z51?3B>@FV9RVV(;RVEA3+Y7P=YN80N5HKS\]RQ[;KX;7+I]KYUN\2X
MXHW"?N?\_QT(C^L5-C%J$]!.OD5N]X5M(A':<=AW?F=7M1V';I_:+<X:Y'&`
M7(U=CF6U!];%H:]ZR.H+UK?R<@S$RZB;XWHL.W`[%"__I37#2%N7RKN4:(ME
MY&FRF>6E9#^.-+Z$X2^E\-</BO7K&=JC`L^ZY_'LZ\8Z&OA+8T/\3&>JA\^+
M\3!U$7Z_R/O[%LZWRSJMOA.!CS&/N/?*NF'U0*'WHTQ#^3\3:-=U;[P<7P3/
MC8S&U1)\;7EJUS!G6$.>%N*^.9`NQF%/9.48A+4[61^>+SUI-_(OV(V'9FZS
MKV+E>(-5[H9[Y[.LG&@$>A[JWN;.A3H"G2WC\S..UZ^Z+\&_P6S916,,OE4.
M]__9-;0-1G/TAW;QM/JRD)_;H!Q+VUA?;);5MW0JX'C\UC9^Z=(S6%]B`MA+
M<'^K`N[-Y"ZQOGB;U=<-]\L,UC?J#*OO`;B_?3K<-^#]S&VW&ZP\JI'N);B7
Z/Y4N%?H_D-P;]"]`O?3Z7XVW$^E^UEPGZ1["^X_1/=)N#^5[AO@WK+Z;C\#
:#/;ZF/@EIM.]5%*Y-L763W=\'/A]^R[E?52.S^'^AK*Y55W+NIIP[H!^NA3
N\NM4#][;[4?N2O>M**%K=_:$BONN27>TM_8P.:!?O;H*Z/%/C>,+=O:EGQQ
#:ZM@KYMO>DVY`W0=5N#_B39P.WWH(X>A/S@7.2-\>85@0Z&=MT?=G']7\7]
"X0]*N:NO_>N^)YQ6J*I?\'AP7FU>D?M&;\[:)R%,;%7S^RQ;![@]8V`8%NL
;"]-S2=G9PR^O]-RL;\3RM-%@^+;4KS_Q*!8\X/WY\+]QX+Q=]`M9]$]K@OO
MHGL<YSZ7[L\"O?]LNL?U8D80=A#G56F,'^))TSV.ES_R@9@?QF?<D^G'\-Q9
MT6?U?H!S\>(9OV\YB>[9()_OX??]D-8)=+_TL#A[F7]3?#BPP9IZMQX6XV*B
M?X&Z&9?;BK:A"W+`>I?FCDH->90YSK7@#K+AK'?#M8I0GZ/X.'H01GZW>ET,
M_M;>.O[Y5V8DGN=]T>+&Y]DB@]_G%H\/[PL5]\>Q7#EV:/[R#LLZ+GYS8OG]
M@X-JA_7DA<O`'HP?FC][IM5UX69PX]^Y,^LK_$S+0VMGXS/VY1"V<3N\C]\<
M:UQ'_J8=6MO(TQPUX?E28EKOB/8*S?6AS5$&GAG`\R%GL'(C\.-GGAI<_</-
M@ZNO>"JPZ2;UQL']D_!<`/=//R5TE+OP'.EXTR^GW;%H3Y*OVYK4>^C=<)TE
MR,A19)<U];[QSN#J1QOQW.E)O7LK_11&A;;;']\1]L2-'2^LV1!O6[%OR<*!
M7,7:,[0#\/O5?:->_&D.VFW[&)9'V_'I0<$SRBCV:/.HAWZ*[>M4R..-D-<S
M`-_G96CI;8?[Z\#M%,"#X+8&\F\EFGG^<SS_4WO1+MD!;1'[SY^]0WR1'$WK
M12>#^Y3>1\']#S$L2TOOBJWB.]<"V#J581_&<S]UJUP9[H?O8)L6ZRJW#XBU
M"3<!C[\0/_[9%XYI'?="FAV'\D30;G2^<#;)M@+<ZZP,M%E_[ZNC\H\/67,2
MS?$]72,3N&ZZLZO\^<UB_1>6U2)=&]TOWRR^,1'\/SH??&?RU7?('@(_N#8(
MV^H.]S./?'$PH,F8*GWY2^\,72<V="POUML,\6#?]26H_R*D>S3\GMHL;%MT
MOQ+<'X?G,?#[Q>:(QS+QY@H>:^Z=7\5C8T+]]6+(P_XXU@OH/E4\%ODY[QWZ
MMJAB+2R.<:%MA-\RB'50K;U;!J.Z6@?RMX'6T3X&;KLKZ#]49\`^%VE5K-H_
MI5I7P7ZG99'0;9K:9E+_%>2Q18S_03Z_R\>9FGN1CX-OG*Z!^XV;!E??M%F,
M,V1`3]K.F,9M_F1C:'.->T>L1>3NF<A]U#M'L.,$OZS$/!4WB?-U[3CGP96L
M$,6Q[VW\]AYL^]A*D6ZIL8H7=KX=RLV:=W]Z>RB?#-,W@VQJ(AJ)<AT5KO-P
M/Y3;^MNW.0]"N8X*XT;WWX#[>Z/P#.[<5IQSG23:TLJVPE'YLSIR6S\S2\RS
```

```
M3@/:70OE.WDSMZUT=)L*]U>"6W(S'[O2#D`]XSC""X#8YS_)Y\[::,P`V_/Q
MO8^\7?D=14M_^P>#\W2P=UKC?.P'RGX4EVU!_GX8Y#LYMLJ]\+;0#3"_K91G
MEAF;#_:'03^K@["%:O?\V[R=KHP5V_(,Z@WZ.UR+4FYEU6>RHXP8=A\#X$<^
M3D%QM$`<2@-^/\QPS33_'F\;T*4#?B]MXFEQ>FV!^Q/@]]M-?/Y%PW';`OA/
MT!@NZN]'XC/0MU;*D!:VE[.>0WYB?&RG$?>&"<>PQE;QSJ?>%FU7T'!<U;M/
M5/)<IOK=[!J>JYVC"LH_"?*D0?E/?$+(GVE/T#I[3OMQ57R8AG@W')ZS[4:H
M\V::PS>@?7T#:/+"DX.KO[V)VT-:$=M*`\L7H8S<AJ5GL(W+E?P3`SGS)+R;
MP5KX.E]!@W'Y'7'Q#6KKVQ5M.CD^=)_P]DA[`0F9Y.!WK;E9?=T)T"L:XL_-
M;X-^%.@,>/U#?(T1KK]EN]HS)^3;9SL&GM_>M;Q]11+TP^<.B_/I#QP6XP'-
M]!T1?E^$>?FG03%>A/MQX/C]+P[7[ALU$^C0`_2X8!.W17I0QLY@T_H96[LF
M\OO3(ZY'VH%Z[1E6#ZYQ2$;?ZT-=C^?K=)`.O]DIUGO"OV:K@3V'^LR:[WU^
M&ZY+*\BL/#_!^O^,>YWNY6."*V7HVQ:"'N$N63)@4U\`?#AOG+"95P9]WSKH
MVQ,+[(%DDO$^.4-KO5B"ZJ@@\G`?]/__1'FX">*XY\N7;4.=0%UTV;8ID.8_
M7#IGFW[)G&TLB.<DQOMU49]'YPLSZ3X#]Q]CY9L@[6F)EO[EXQCPQ905VZ[Z
MZD"&]*$2](5(?^RKMOV]%;KG8@'?1'G"_N0;E"_<EQ3UDA?B;,4^D&=-\1<?
MRQR<U0-ARH69J*\$\JVY=PF$0?K@W`R.B;:SBQ[!,5#DW6FL<3?NPX7O,8^X
MA\6,Q*3^IT<A/QW?R_T/7-@3^9_4?R7X1UU^;;RMOSTVDC\1[U3NKZ4?>'<>
M^C\M)O:,&LG_!/+?/J(_D3[2K/`Q+&<SS8FU]%H[A1S?B/O0O#<X[[.@5Q9D
MB_/+,^^*O61JYD>K]*YGAM5Y<.Z2KXL>A]\%=>5+H#\Q:^.>[8!+`6>*^:5@
MS1D_NS[0`9"?)^_D.@#UH]7OFN#=W?'N?&!+5NX#L./M(?$FC\D'MB6&;8"P
MMT"\WXU?+/+4)?(4^]C&/2<.S5/FF+RP546Z[_6+.0R1W^IW[]`9]['1A:VX
MM\1ESX1[2T#^J_V^VB^^4;\"XKH[/G?8<JRO*<>Q5>78VB_L5I'/8_-<1\5Q
M\:,*6Y_M%[;BM$:P'<!VG\_.W]TUALT[+<S[L57TW$CY,2OF2"KS\NVA>2E5
MY^6?^X4-(O(Y(=\5Y`5H\1"\N_`/@ZMEV@M`Y'="5?IK^[$^Y^3!GINDLO-%
MO5Q`]3)GXY['<;[F\*#Z'+P?+OQW*/QS//RLZO`7;=SS$(5_4JQW&V8/G*&\
M*_8VV@MV88;VF>+CA<)VX'M-)4.^G5#5WU[3+V3-!;CF$G0GULCFW0+Z-8:]
M!>JA-5[8RA(%T'7N7A/D\\;#0E_GW]F-#O>"`CI/J%I3>!G%G0YI?5SX_F)Z
=SK)[X2!>\`T\[T#!,V."WED)OG%,836,86MR#,;XPF^KY@HTW%A7V^0WQ:2
`1OCK?W=\4#.1OYP'<%'R"_VE5CW_4AO:O?3*\K(WA^<MV58.V*8O05!1YH$
NG#+=:#776.5Z1O?,,W\8#AV#O286%471XG\3+1#OJM^'^L7<T0,\MH,-%@5
SZT1Y9](Y1+[?1VHV',4V_;GGZYLVQ.KZFCG6R+-64$_>HG@'3X^$V_B^ZET
3UB[)FA;C^VLS']35;HO45RIBKAR\6!/\TF]$_^U(A^9ZK"_I+`=89NO?K_I
K4H[I?K=S]_B\Y$KD];Y?7S?2&MTGY`+8L[P]L-B+I_KC>@'U__@M\RXAAO7
,:/[G(:^8+]"S,=?LQ<?M[^`;W'>$-K!!A[/]&X^MU8<Q3;8\=B*6^*Q?B<A
OC6]"=XC+6S0(:E.@:8MG?9.,0>&8^!NM,<0T+=9[#$TQRJ[EUCEI6\)N;4P
/CM?=([-Q\"-S4$;EJV8P9KZ3P6;!NR)YX*X<0^<"RENT)54)]HS!>C?G.^"
L-V7"+WTJK?"^36@/:6+_J>+_9><M_C:6-Y6JN(I5<=S^5M\K2BNDZZ16[A^
%3$9SOW4KF^.+2+[1DJ4<2V'WB]T>]RG]9?PNS%N]17G/SP1[)=QR*=7@@RX
:XEH9V?TA^,(0+L6E/F=.(Z`^_,N9V(O>0A[`]A*5^/Z_0MC;%58_[D&W)MO
93/N07N,58[MC)4S$W&=`>A^%P,>`]@*.!80UQQ,!!P+V`KX8<#)@+B>ZT3`
LP`[`,\%_##@9)Q3!#P1,`W8;?7EKF)]:Y>POLSYX'8QN%T(.)>5U_Z8]04R
J"O&=B5/393Y\QQZ_E"B[)QIE9=U@UWF6'U+SV=E"[_IR!V?CS>P<JR1E3_W
[N#JM=="_#>POC']8GT*XWLFM%3)D[>!GMW(3V_1.!'CW[1&]C_?,Q;WB&?5
^_\(]UVL4!W?2[C_[D8Q#OGR+\0WH=Q?J87W?R[4W1;P\\8OA)][A)]>[B?9
&OKYG^#G$/E95^DG$_GY!?AY;J=8%X%UB#2(%2:8WP3_:`<$WRA^=M?@:JCK
5?OYON.8Y]8\TK%]^LQP3N;A-X<;9ZF5[9^&<'MI7Z#\6S2'V&&5[WQ+C(MA
_:YX2^3[P#M=O+Z_\9;@24&'UGS0%]P):>)[=HF8AQ<TF%1%S]O!SRNHYR:"
;Z%!SZG8#^K2?C$')VA3'?:F-X5^Q^>N*_;$.K\?U]-1F$)UF"O?%#8[?7==
E99QA+0^]R:?O]O5A&=([.SB<\E@`W:^_Z[XEB5S?LP4Y:\.]_$WCVP+;^=M
F7_;ROT?_=;P\W<X1M<$<H.W`VMF&>S)#:@[<)ZP$N5@_8C\'"\#T;JM:BZ]
D\K`OW.-!6MMFGIQWZG#[XBY?%'VZG!3WQ2VM<7$O@2!^XXWAY_GXVV8'96?
P8XRD3^AS9KQ\<S$=2"QVP"7P^\.^*W$_7C@'8N9\0;X-<)O5LR<B>L"YL7,
V'CX38!?,_S:X)>$WZGP.P-^!ORFP\^"WVSXS8'?)?"[''Y/03Q%^&7B9GQ!
W(Q=$Q=U4VBKJIO7_QS1FM.%\?&[B;$&"-,(O_'PFP"_9OA-C9M#ZVW:S7&^
/TJFHOZ^_Z;8_R2(+[[`-N8?2EQ?P/UA&]@G^7[.(--QSZ-N_,X0_%@+V@W^
'03$TSU5?*.ZXLW:<0N<$TPFI^3CC8]M[<@F0(ZMW!H#><C;.NY?TTCW]\#]
>)"3\<>VQB8@%K;&FND=I%<:7+NU%'MLSPS:>RF7P/W@'MN::'ALZQ??I'5/
D$['!+`Y:6];O@YK\<SR3)98@=\2.13.#N1DJ9JVW_GS"./-M(Z6ST=W3,F+
^?`I^;LA+B>.Z;7P[Y1<Z'\P+=2!E_&QSM9.M&_.@_06=N,X>'/_THIYR"/M
T<'KXJ'S^ZPV<;X+;S,/C>[#>>P<I#EOJY#C8HY\2G[:O8%N:)5?;8`Z67GK
Y'NA77W]"9!GT!=^!A'ZPMXG4/Z`WG&B<./WT"?^"M]#GW@;8B<K9X-WT#]^
#MV@?]R(J+'R+Q'/8N7&P,^YK#P.W:#O.Q81^KXFQ`M%?-P/]*6WHQOTI1]!
_!0K+T;\+.-[N%DQT<_%+8N?MP#ZUCS<S^M4'!.!.O\FSBLT6GE^#@F>W9%D
SW'>FMY0#O9&#^HEMICKB[NZD\?G40\U7R"[D_PQ2UZ!?MI9TT7<GW5\/@'H
9H[/G_D"ZEM-XT`G7/]"QPGY-@@S:3&W6W9MN._X/(Y?!W%LCS==Q/=2!?TC
">$6@NZ#WY7A?D!/'L;S1?Y">L7C\Q,I/9[W32?D>;J;3OSKTOU[:R!.Z6*:
V_^:--GD_'O/BS2Q?;1,GU56X^S1?1#?AAV)YX)T6\'O"\.FVPKISA]X[?D@
W9:_+EUK<OYW$(:O!2Q^WEP&LASS,&(XJ#N^OJ=X8OZ9YX=^BUK='W&9!&VE
`_H"[-]0SLV?FKB>S[U!^%?;1'MX<4O4OKF\OSQ1#N7HY3.C^\?/[\/U!+S-
M%2?GIX$_[.^2C:RO9GU4L.\(RI/<9+Z>*)`G7WT#Y-(2T?^AWLSW]1YF'P@N
MXR#]./27T+R!'L4]70N2`SQ.2#^([PMO"+LXW+>_?M6O^E6_ZE?5-?>B2^:[
M64<U)<E-GKP0'QTCK3FN:L/C6';AC=<D4ZED6DH9W'\J=5;*/$N1*\.G==N3
M'"_%`U2XI_QTVC-E<+\.'[/9E"IKJGWRXK'BO>/ICJ.,_#Z==30SZ\HCOG<U
M*:O:^DCO4RE/,GW?&,O"9UN2)&4D_UG)E4PG+07^?4EULF9*#IY=6384/:4%
MSXXA>Y(FA<^^EDTKKAGZES19U92L%SP;IFOXFL;]7^M]:>&7G1NOO";KW?"1
M6=["J^=]Z;HD>-*4^5+*MA5=2@M"QAB;L>^;#^QZXX6;/[GN[_[93;_^J;W?
M^$9?TWUC?SKU"V>,^?DW]SW8>.$7[VS]Q:\N_]'U.W__Q?OF37SCH6<V;/OV
M'^[8>>\/?G_"$W.^_?+KF__8<\_Z#7]8>^\)ESYZUWG?:?WSVO/>^(\SQCQQ
MVKS3_B6W\P]KU0>^^-US"F?W?V+\`T9ZY]0/[K[<OO^<\KL7Y-9[XY8?\_;B
MLW\]<]_/EMW\BP?7-CQY\H6-!R9_?/T5+4\\^-.;)]R?N*U4/'W#(Q/F_O'%
M^?_^XY]?_;M+NH]^^2>+EW[JG$4SBR\]^`]7%5Z[^NP;_^V6<R]>=FWQ\D=>
MNW][_@L_?VKQ"8?/?FO_8P<[?JY.G=CS\,`;FVZ+G?N11P?&?'IJCY+]]3W+
M;MXPJ_O]ZY8?=?//GGZI`<B0J.0K2?5E5U%.7HQ/GNJH.K`:KT9!][0DVVHZ
M%=:;;QJZK:JBVM.JFX)Z-,63DI+DK*LZXDF5?=65_=3)UXVM2,\T4DI:]AS@
MFS,J?F.#]X[NRKKM^N`6.'-WS_!36<7F.1/M1/52*4,.GJ64G]*TM!KRIZ;)
MFN^9(;_(J:R;-0SB5SUEV*YF2B&_ZHII2H82M!\_JZK0;F5!EBK?E!]9\E*F
MK`L/63GKN%G3$%GF=#*\M*]+V9KWHO&D'2B,0_%]3EI\\N*S^+^3LU<DB=_3
M*<]UC&1$/NXNIVTHA><&Y<H:IIUU?"EY<E:('\55-"-%CU[:2?MI10_>ZC;0
MWM0H4IZ.[Z<\G<N-JG1<SY4<R0C;HR:IFFO+SG\W'<>37<?4O1$"5$=G.BE-
M]Y4T/:8EQ39`P-"C8:C0^I4L/"9YU&Y6-15%&3[QH)RJDO:];`T]W;0-`MO0
M0S[QTE`YIOS?IB>T`5V21@KPWRPGKP$WY6BR9(]83)%_-RL;?EH:6DY%-Z&M
M^NFP/E)9V=?T4+YFLZ:6SKK*WRK?NFK(FFR']>`Y=M9SM3!_BF+Z3E8+^=Y,
M9:6T9T?]D2OIT&2RP7-:<]64YOC!LZKH65WWP_[(!`9/&Q$]9,>PH8MSAM(-
M'$W%1^:J=C>@;[)E.SW47;5]U_95.XQ7DWS/]T!T)(>VMV15NS;3;EI2Y:'Q
M.9ZOJ[X9M#O;3^E`3>6OJHBAE)<\0THKJ;^VG53E3P?E0-/T&CDA&ZIB9G7W
M_U;^*OB[*G^.E,[*7ACU?RD??X%A_U*V!'^F-<_0I+"_@0SX:4]60KF2SJ:S
M%?J+XVI:2E-"^9HUY7364"-YJTFR)#MAOYNV32TKR>%[$SH4'_@JE#NRG];3
M<LC_CF?[ANN$Z8->JFNR&89/@SS(*DH-OQFJ#G)?K^%K:#Y2VDG5RA5/]EW%
M#M.5%<_U-;>F'?FV;:NZYM:FYZBV(>LU[<CQ?=EV0GD@FP;TQ]@.%E>W3]!&
M'#T;R@5-]7S-=#1H;YPL6=U6)`>;3Y*=P+["GASF+\6^SY&%+I>Q:>PVMG68
M/XT]S)&%+E>P2>Q*]J-A_DYGRSFRT.5"GF]I,?Q+C#YJ_+$36]JFG-C^H5,Z
M/P)5:TZ?T7W>[(LN_N2\3W_V"A9O&-TX=OPQ$R8VM[9-GGIBLN-#'SZU\XPS
M@1,UXZSIYS*VC?\A/D/WP5\3.YKC9>P3%?24LZFTK@Q#3\$OON+8AA+J3[XB
M.XZLA/)1=57?U<T:.6A[JF;ZME;3K^J>K?FZ.=3=U$&=U(P:_M`AN93AU_"'
M:_II1T_5R!T;U'^PE6KR`UH+\%FJII]/^88BZVB(#>$S0P:%RO.'NFLJO`%:
MU\1C@NJF&37M!NP;/^76]A-R*NV"'ER3;M8&>TO+UL1O^JZ3LFO;B6F#F/7-
MFG:9U7Q34O6:_*1L,!A]LX8.4.U93;=KVINN.[;I9;,U[HYC>(Y90Q]3<11%
MEVORHYEID#)^3?Q>"O0_2?%JXM%DSP`%MI;.$+]FJS5T]DS7U-P:.ILZ=)^R
M5),?W_$\W<S6RJV4XGJ*5%N_MN]IIAFV`\E,9T%0U]!%4CT]Y>H^:?0N"%,'
MZDBH^R,]5?1C:=6#WCY;4P['563?T6OR:Z9!&]+LFOI4H",!5::F')+KZI*1
MK:D'.ZT9_C#^4VE?@@ZDEA]!/Y8\LR8_H.:"]C4,_310$[24$]AE6MI4/<T.
MY0AHV[KL&C7Q&4!SW_0]HJ<*W;$,-AWWQMN1Y_NZZX?]H0/\IQF>'J0#?.%!
M+U^KOQF^(MEVK7V34I2L8]?X5S1%,M24$<8K@]&9=93@&;0&4)3EVOI1=#?K
MF37\:F:-%/2I-7ROIQ0UZSLU_N5L%@BDA?0#F0):EQ_*;57WL6%HP;B+8?N@
M)<BDR-@>B&G5H\[/E34%Y%Z8ALA/"D0VRNNL$%LI*07FI.#45-;3P8@VQ9,D
M@<FJ.L3%JN&DTY+IB2>0+&E;2@'Y`WJ"5:_)M711LX;F0:["^E=3BI=R0GX`
MT>ZF?=<0Q0'9#7Z%%D_M)*M[D*U:.9`%>]2HT*=D50,C)=*70!J;OI\=B9\\
M(RVYLA_JZ;:<M4%G<X>.N]F.["E9PXG&$1375*-Q+*"XK(+1&#Z#S6%`)Q3J
M=:H)W:8>ZH7`<F;:B.P8X&I(UPGCTWW;`=*&^I3BZ4Y6-L/PGH']K1[2$XAE
M0II:D%]@04_*&E+P#%JIE,U&>B)T@[[G:,I(=`G\A_')*1"X62-XSH(-I6A>
M2#=32YFV9XP8GP]*KVGJH7])4WS#D!3A'3@&%`O'C>K;577'@ZH+Z]])FZZA
M.:&<\"10+!0[E"_05F53"?G`MAW7`,4ZRK]C0(L-RZ]EH=/RI#`_KI%VY90:
MT=MQ?,=4TX+--04Z,]L/1L=T&ZQ,`]L.\4@3.Y6UL&4C_,UBL^'_G!'?+V-=
M[%QV-FAI<X_@YV_W=QF[_/]H?#?"[V9V"Y-2:5E1-=TPH;*RGD^Z\`4M%[1-
MN>""$SL^+-3A"^B:3KHPR7U349UTR.\IUP'S,Z6&_)X",Q`D9"@GTJ!AN+8=
MOM=LD(IZZ!_:O2*94BJ,#V2*[+IR*"]=!<2EZX?Q@74H@>0.PVM@A9G9;!B_
M*JN^IX;\ZH).J+IZV'X,)9V57#_TGU5]6TFG0GYW9$W5%3\LGZP8NJ0+_T=5
M]A-IE`).F$]9@^[/,<)X#!]Z=#5JQ]`10(]CAN]34MHPO:B<:;!-H1Q>6"X_
M!?JF%]+%2WF>G)9#NLF2JWFR'M(53"Y3-N60+FE=SH*:%CZ;<E;-`B&#9]TW
M-1`<87C;3NN2E@KS8Z<-23;4,#T0:F#R9D/_/AJ\IA32&?1.75$B>D#2:575
MPO*#4B794CK,CZ,9;LK60SJ#-6S[KA?6DYZR)<]QP_`J:H6N$>9?`JD`]1*E
M#^8JZ!.A_[2J2+X2E5?6?-!RW="_#T)4245\8^J0(G1_8?W8H/%X7OAL>`8H
M2'Y8?Y*<2J75;)B>D])QR".L+]]-J7I:"<L#1K(JN69(GVS:=,#"B<)[T*W9
M9IA?7P?)ZT3^4]"H5##UP_K0TH8!O[#\H)2D*_)KZX9CFNFH7?JHSD?M1K)!
MJ33]\+V?3>%X>SH,;WA94&W#\NJ.ZVA@_(;Y!WW+D"/^];24!^VZHC[`;M$\
M/>1_%0=]G)"_9=#>%5N/VC7HW0Y421A>`J5$RT;E3:4<0S)#^F;UK*H8:IB^
M!-H(L%3(GRIT'$K6C^@/#.&`(1+Q5]8S[%28GIX&BT#1P_R"+@Z*3U0>8#])
M@PA"?O%DH&Z%?P/#1W)/5J!GE-WP&;1:1P5-(JQO7?>`'R+^=\R4ZMD1?^!D
```

```
MH2N'S[X,FH^4"LOGIK,NR-VPOO1L5O)!EPOIX?JFIZGA>\U1LZX">G^0/UE2
MTW+4WGPY);E2Q-^@:LJ*[D?R#2>:[(C?TSX./*9"^JB@:$IVU%Y=/PU-U(CJ
MPS#2'E1*V)[`8%0J^,<`;@(S,:2GKI@0?SK*/S"G:Z;#\J6\M&S+;LA/T+1P
M?C/T+Z=<7TLI87R^I)DIS0OI[2N@=SA2U)Y!."NF',8/I7=ER$!`K[3K917-
M,8-G3_+!_I+#\ADR2'M?#^O;,:`3]&PO\*]!AVH:2LA_/G1DFNI%]'-=UTF+
M^ECRE2]\:9$7H-"E\+)!`(?QIW0/)$H8OVV`.'!D-WC65;33L]+8).4?;"@3
MC/'@&7@)1(IA!L\I6?9D.Q6^][RTJOEN&)^3`FT=.@A,_Z,+G2NO_6B0&WS^
MRL)%WA<_REVONV[^=3=\R?46+I1IUG>D]V'!.#T<(+<N1_+9!'GA1?5A2-!:
MU:@^/0W\NTI%_VNZ(."C_E&74L"?4?V@V:`8$3_I+@Y[AOYUU[5U6XOZ/S2$
M[$B>@W&NI!PI]`_=O^E*:A1_&CI@U0[E&;26-(BSL#V`O>2`2*OH;T`8@@D:
MT!?,Z93J1W99UD\;H)ME(SL-&IAG>%%Y?-\VI3!]Q?%ML.$CO0GZ#T=50_[6
M%$<"U3JBIYH&@]&.YO-!G,I&-'XN22G5`9$9CE=*GJYK1D1OD%6^$O&_XX(X
M2T7T`U:S)2W2Z]*F8:8J[!A0#E205U'[!=^@"(;M!W14'Q22D%[`W3*4*6R_
MJFN"Z1VU7Q"WOIK-1O(0.C`O&^E/J13()\\.^2NK^"8H=J$\--,@,%)1^]4T
M-"0CO16T/=GU*_0=5U7!DC'"]JVYN@L2(J2':4"%1?7A@7QWE4@O]3T)Q%%4
M?AU4<E"Y(GT%I2'TZ.%[T_5U+](?LR`P);6BOJ&MIK(1/:#JLCC"$J4'#%S1
M7Z1`6INR'.E;0%LG'>5?3ZNN85?P+RB4FNY%\DNSU;0=]6\I'01@6@WI;4,!
MP3(/PYN@OCIZI#^J8*>Z1D7^%`GZ3RUL/X;KVS88IR%_9!U33^EA?!(4SO6B
M_A.L>-VOD*>@2@(_NF%^P+OLN9%^:&95TTA'_;<'W2G0*&S?T-V8NB9%_3%H
MZUD_XE_0=_0T:)!A?)IA9/U(/\_:62!(I#_X61S`-\-GW752CASI-XJ2DCS3
#O.#0T>&$]DI"K`SZ,N1?'-!&0-#.Z1_UH`>3(O&(1P@7U:.]!OH'64UTN>A
:X/^U(_J#\P%S9<B^\%/.V"^1/:"+6456\1'ZZ8"\4[]FVF`816U#\4$!4Q7
@O;A9%.^ED[YP3/H2[(#?7B8GBL9NAOI8V9:D].@%(?\Y*0\T`<C_<LQTF:V
0CXX$+T3U3\PEYV6W$C>&9X*"GBD?[G8X;N1OHSK5621_LD+1;D^>O+",Q"]
:]+S%RZR%WE4WI.OX\[@GJIRQ^<;O"]<"9W=#?A\P]77P?L;KKSV"Q&=($DG
;1MAOL$,1TO<9?6K?M6O^E6_ZE?]JE_UJW[5K_I5O^I7_:I?]:M^U:_Z5;_J
5_VJ7_6KYEK<==SZOW4>ZE?]JE_UJW[5K_I5O^I7_:I?]:M^U:_Z5;_J5_VJ
7_6K?M6O^E6_ZE?]JE_UJW[5K_KUOW<EUM?G_^M7_:I?]:M^U:_Z5;_J5_VJ
7_6K?M6O_]>O1*G:_D\4Q7/7&%9DS/I;9.EO>D&Y<XU0;BQ_X_^'Y:]?]:M^
U:_Z5;_J5_VJ7_6K?M6O^O7_YH7V+N/V;I+0(LP0Y@@+A$7"$B%;2N$)+<(,
88ZP0%@D+!&R912>T"+,$.8("X1%PA(ANY7"$UJ$&<(<88&P2%@B9+=1>$*+
,$.8(RP0%@E+A.QV"D]H$68(<X0%PB)AB9#]/84GM`@SA#G"`F&1L$3(OD'A
"2W"#&&.L$!8)"P1LN44GM`BS!#F"`N$1<(2(?LFA2>T"#.$.<("89&P1,B^
1>$)+<(,88ZP0%@D+!&R%12>T"+,$.8("X1%PA(ANX/"$UJ$&<(<88&P2%@B
9-^F\(06888P1U@@+!*6"-F=%)[0(LP0Y@@+A$7"$B'[#H4GM`@SA#G"`F&1
L$3(5E)X0HLP0Y@C+!`6"4N$+$_A"2W"#&&.L$!8)"P1LAX*3V@19@ASA`7"
(F&)D-U%X0DMP@QACK!`6"0L$;)5%)[0(LP0Y@@+A$7"$B'[+H4GM`@SA#G"
`F&1L$3([J;PA!9AAC!'6"`L$I8(V6H*3V@19@ASA`7"(F&)D-U#X0DMP@QA
CK!`6"0L$;)_H/"$%F&&,$=8("P2E@C9&@I/:!%F"'.$!<(B88F0?8_"$UJ$
&<(<88&P2%@B9/=2>$*+,$.8(RP0%@E+A.S[%)[0(LP0Y@@+A$7"$B'['Q2>
T"+,$.8("X1%PA(A6TOA"2W"#&&.L$!8)"P1L@*%)[0(,X0YP@)AD;!$R-91
>$*+,$.8(RP0%@E+A.P^"D]H$68(<X0%PB)AB9#=3^$)+<(,88ZP0%@D+!&R
'U!X0HLP0Y@C+!`6"4N$[`$*3V@19@ASA`7"(F&)D/V0PA-:A!G"''&&!L$A8
(F0/4GA"BS!#F",L$!8)2X3L(0I/:!%F"'.$!<(B88D0#XOGX0DMP@QACK!`
6"0L$;)>"D]H$68(<X0%PB)AB9`]3.$)+<(,88ZP0%@D+!&R?Z3PA!9AAC!'
6"`L$I8(V3]1>$*+,$.8(RP0%@E+A.S'%)[0(LP0Y@@+A$7"$B'[9PI/:!%F
"'.$!<(B88EC8DLPWY=<3^$),X0YP@)AD;!$R/Z%TB>T"#.$.<("89&P1,C^
%WMO'MS8D>?Y85HUPQ9V;'?/]$ZW[;&=+=4,P2H"Q$6RJJ3J`$B"+(QX-0$6
2^KNP?D(8@I7XP$%LEN*@->R.;XOV36^X;7L&M_M7>UR]D;$*J+V'\?Z#TW4
_HEP**+FS_U#$;7_.-;YR_R^"W@XR*XNJ7?R&U)]B/<R?WGGR\R7[[V_`O]@
#,R"7;`']L$!Z/FK\`_&P"S8!7M@'QR`GD_@'XR!6;`+]L`^.``]?PW^P1B8
!;M@#^R#`]#SU^$?C(%9L`OVP#XX`#T7\`_&P"S8!7M@'QR`GC^!?S`&9L$N
V`/[X`#T_`WX!V-@%NR"/;`/#D#/WX1_,`9FP2[8`_O@`/3\+?@'8V`6[((]
L`\.0,_?AG\P!F;!+M@#^^``]/P=^`=C8!;L@CVP#PY`S]^%?S`&9L$NV`/[
X`#T_#WX!V-@%NR"/;`/#@37YSRB':]_&YP'5\$M\`@L@1WP''P,/@$OP*?@
,_`Y^`*<^Q6$#\Z#J^`6>`26P`YX#CX&GX`7X%/P&?@<?`'.?0WA@_/@*K@%
'H$EL`.>@X_!)^`%^!1\!CX'7X!SKR%\<!Y<!;?`([`$=L!S\#'X!+P`GX+/
P.?@"W#N&L('Y\%5<`L\`DM@!SP''X-/P`OP*?@,?`Z^`.=^%>&#\^`JN`4>
@26P`YZ#C\$GX`7X%'P&/@=?@'._AO#!>7`5W`*/P!+8`<_!Q^`3\`)\"CX#
GX,OP+DYA`_.@ZO@%G@$EL`.>`X^!I^`%^!3\!GX''P!SGT=X8/SX"JX!1Z!
);`#GH./P2?@!?@4?`8^!U^`<Z\C?'`>7`6WP".P!';`<_`Q^`2\)^"S\#G
X`MPSHOPP7EP%=P"C\`2V`'/P<?@$_`"?`H^`Y^#+\"YOX#PP7EP%=P"C\`2
V`'/P<?@$_`"?`H^`Y^#+\"Y7T?XX#RX"FZ!1V`)[(#GX&/P"7@!/@6?@<_!
%^#</X?PP7EP%=P"C\`2V`'/P<?@$_`"?`H^`Y^#+\"Y?Q[A@_/@*K@%'H$E
L`.>@X_!)^`%^!1\!CX'7X!S_P+"!^?!57`+/`)+8`<\!Q^#3\`+\"GX#'P.
O@#GOO$S&O___6^#\^`JN`4>@26P`YZ#C\$GX`7X%'P&/@=?@'/?1/C@/+@*
;GU3Q//O'X&E;R+]X#GX&'P"7H!/P6?@<_`%./<;R']P'EP%M\`CL`1VP'/P
,?@$O`"?@L_`Y^`+<.XW$3XX#ZZ"6^`16`([X#GX&'P"7H!/P6?@<_`%./<M
A/\MD=_K\^`JN`4>@26P`YZ#C\$GX`7X%'P&/@=?@'-_4?+;X#RX"FZ!1V`)
```

```
M[(#GX&/P"7@!/@6?@<_!%^#<;R%\<!Y<!;?`([`$=L!S\#'X!+P`GX+/P.?@
M"W#NVP@?G`=7P2WP""R!'?`<?`P^`2_`I^`S\#GX`IS[#L('Y\%5<`L\`DM@
M!SP''X-/P`OP*?@,?`Z^`.?^180/SH.KX!9X!);`#G@./@:?@!?@4_`9^!Q\
M`<[]2P@?G`=7P2WP""R!'?`<?`P^`2_`I^`S\#GX`IS[EQ$^.`^N@EO@$5@"
M.^`Y^!A\`EZ`3\%GX'/P!3CWVP@?G`=7P2WP""R!'?`<?`P^`2_`I^`S\#GX
M`IS[5Q`^.`^N@EO@$5@".^`Y^!A\`EZ`3\%GX'/P!3CWKR)\<!Y<!;?`([`$
M=L!S\#'X!+P`GX+/P.?@"W#N7T/XX#RX"FZ!1V`)[(#GX&/P"7@!/@6?@<_!
M%Y)*2DI*2E\-O?:/U?M8E)24E)24E)24E%Z^MJ[=^RVZ_[/U+?"ZI)*2
MDI*2DI*2DI*2DI*2DI*2DI*2DI*2DI*2DI*2DI*2DI*2DI*2DI*2DI*2
MDI*2DI*2DI*2DI*2DI*2DI*2DI*2DI*2DI*2DI*2DI*2DI*2DI*2DI*2
MDI*2DI*2DI*2DI*2DI*2DI*2DI*2DI*2DI*2DI*2DI*2DI*2DI*2DI*2
MDI*2DI*2DI*2DI*2DI*2DI*2DI*2DI*2DI*2DI+25U&O]7_C9X[?_UC^
M_O^N>?I?2H2^9/%T=[\NV?_ZEQT9)24E)24E)24E)24E)24E)24E)24E
M)24E)26E7Q*]-G#N/U!24GIU^K^O>;[A\<0X&1@#LV`7[(%]<`!ZO@G_8`S,
M@EVP!_;!`>CY#?@'8V`6[((]L`\.0,]OPC\8`[-@%^R!?7``>KX%_V`,S()=
ML`?VP0'H^8OP#\;`+-@%>V`?'(">WX)_,`9FP2[8`_O@`/1\&_[!&)@%NV`/
M[(,#P3_O>NW_4=<?)24E)24EI3]?^H?7/+]-X\!_^"WP.A@%-\`T6`1;X(?@
M1^#'X"?@I^!GX.?@%^"U7T'XX'4P"FZ`:;`(ML`/P8_`C\%/P$_!S\#/P2_`
M:U]#^.!U,`IN@&FP"+;`#\&/P(_!3\$OP&NO(7SP.A@%-\`T6`1;
MX(?@1^#'X"?@I^!GX.?@%^"U:P@?O`Y&P0TP#1;!%O@A^!'X,?@)^"GX&?@Y
M^``5X[5<1/G@=C((;8!HL@BWP0_`C\&/P$_!3\#/P<_`+\-JO(7SP.A@%-\`T
M6`1;X(?@1^#'X"?@I^!GX.?@%^"U.80/7@>CX`:8!HM@"_P0_`C\&/P$_!3\
M#/P<_$)224E)24E)24GISXO^Z;6O?4WP6^!U,`IN@&FP"+;`#\&/P(_!
M3\!/P<_`S\$OP&NO(7SP.A@%-\`T6`1;X(?@1^#'X"?@I^!GX.?@%^"U:P@?
MO`Y&P0TP#1;!%O@A^!'X,?@)^"GX&?@Y^``5X[5<1/G@=C((;8!HL@BWP0_`C
M\&/P$_!3\#/P<_`+\-JO(7SP.A@%-\`T6`1;X(?@1^#'X"?@I^!GX.?@%Y)*
M2E^F_NFU7T?]!Z^#47`#3(-%L/7KJOXJ*2DI*2DI*?T2BMYW)\G`&)@%NV`/
M[(,#T/.OPS\8`[-@%^R!?7``>OX2_(,Q,`MVP1[8!P>@Y]^``?S`&9L$NV`/[
MX`#T?`C_8`S,@EVP!_;!`>CY-^$?C(%9L`OVP#XX`#W_%OR#,3`+=L$>V`<'
MH.<<_L$8F`6[8`_L@EP/0\X?P#1B8!;M@#^R#`]#S
M[\`_&.S8!?L@7UP``'K^7?@'8V`6[((]L`\.0,^_!@#,R"7;`#\&L$!Z/GW
MX1^,@5FP"_;`/C@`/>0)`_\`'8V`6[((]L`\.0,^`#Z/GR!=#.E
M8V`6[((]L`\.0,^_[&"&T@/G`+/0.]X-[8V`6[((]L`\.0,^`#SOZRRYO88"?[%O'#'
M#UP_V`6[((]L`\.0,^`#Z/GS7&_V`6[((]L`\.0,^`#Z/G
MX%]Y@"?V`6[((]L`\.0,^`#Z/GS
MSW!19[5\NY2IU$N.'+7G#OT6(;2X%]YX?Q#^$3?YT^``'R#AT`;XW?G#S1^R`
MCZIXOT$UG@G3C)N^\\/:&PL3LGE[;PM)*]^\:4^:$6+P1X[\)#\;B;7#+>F3
M4GJS;`1`$C'*M'E7$?3!QH(M"YPY,"WMYF]NQ59Y*3(WC!B^-3$G,I?*`Y<,
MX$X=B3<]C";=B"3WPO^L:+6A#""5CYGONX9#>W"CB9`!W6&\;V7U8Y9O'Q]K
M39D4NZ=\4\L]M`Y],*THK*BPFW>',E84$3H`45*4@P7>Q?O,DBC4B]JB/'F#
M]_&+MAZF7G?^"CM_1IP_HQ071V=#W>ZBV?TZ^R'9%R[:>D7[>;V3:_#Q$ET\
MBN7F(EUBC#_E1257TIP^G!>41?&;^F?C&H*::@ZX,L8UTU%?1Z^G!)F7]%=`
M7$]Y?:#"3.XFTYF=^%9RW>9`))KJF)$U"YG,HQSOMNCX@MVA-LZE-NS4F3:'
ME]]UNY":GJR+Z*@G>^8X/-DNKM2L3L,TGD0U$XX<XX=9DFH;5(Q-[\W@:8B'
M8_=F&W$XH^^L'78?YDC$Z<&L/C:W]L&)T[6MAIFN1X8+J`.)!^F#>&8G@>O5
M.!_(S"$?U$79*@3:A%L6X92S_HQWKHTZ=XXE$)?-Y,ZZ:^3-P<6(0Q%GX=#L
M2HRVP,<(Y'G1%I_[\<S>YF8JP<\%Y7\\3A]XO^P!^B]85YG_Y8XK@=0EPI@R
M_PN'(RO#\[_H2D3-_UZ%9IR]+=WPLALLOKG-J*)H3=8IMT[XWY4*T\_TEE9E
MO!)5<JURG5_CZM4JGT;JW,<2][G$F^FC^D.-&J?TUCK16(O/-0LY7?,&2I5Z
M/E<QFVB&`LCP*N8-Y"KE4HV%O2.G[HA&??*H0'A3CI^;6LN;V-W8/]A;]XUX
M6.#75(J)N`QD^#CLH<YK.$^&J.KDE$>2)K^M<H['C-E<E&NLP1M!K<6^G]@Y
M-*.;R8BQ3V;8HA7K<2YDY//-AR+RD9'(C_.W8)1"26MEJ!W:(M_4"EKY$8^X
MD:U\\%"OBO+B)<$:E5R!GRS3$($U^9$,']!Q4V1ML]S4>=J.]@XV:*1G+QM6
MUL5O*C@4*A_=!%A*T_ADO5P+%,20MLB[AW)%#XCR'LX?([*C^6*<D?E1X1T]
MURGO=^]:,>2JUA^)$R'>(8O+KB/[HF.SSS!O9IO,SZ&,HU*G89ON\@H%W:JS
M>JURQH/FEPB>0$I_4ROQNBUJ!ADKU'F)\,NN:YJMD*Q4NYR\XT@?3_B;COFD
.,LO@$;JG2[XM<V1$<M3ZI$C*_C@9GQ&%$[*E2+E@7;*KZ[L0="H!O2;/<I5
MVNZI-HR.EK1QQEGS5\;&V'"_\,_ZQ?<KH*M<_\T+PXQA3+G^!Y>#RT/7_TAD
M5:W_OA*]63ZN\<)DF9UW,QN)S53F7L;[)C]0KFF.8^9!/DQ>2Z93+'+;?DB.
MG'D/?2QTRRC:H-?RF>$3%=_I`O-98^Y3YK=\+]A</AIU>=/NTG1J]!TU7CD7
MV`]%GQ)H\0K,Z,@B^YWC=JU`PY*W<))[D(X=-IS^Q6Q"^@_XB=XW>9]4/F9+
M-YB17\@8,<;YL@OQY]!5VG^^7N>%,OL48$K[7PV&A^__1((1-?Y_)7(=[EL'
M<[EFX827]>3[/+R1:LUFN]'RG_"A6T5KRCL^&`B$POQ/71.MD`7D0ML;N5/N
MPAQ"-#6=CP,D[IAKY/Q'N\$2VQ$:NQ4>VL:)IQ$:)XJCF5:]D=$J$=M@2F\L
MDHL1.Z$9[(2D';TI[-`!IZT=7E\8-0"6:U'4I/.FCB'<>KM)0\C$MEPKKC:,
MH5T\?K!^C_=DB>V,X2L?T'[,6"@2"LN1;KYB+9Q2"#SP#,]6&G#261;PDELK
M>U(B6=II06N(K'W$L[C>%$-4D5@:KE/F^1+;81K#R6S(GS%>?@N.03=/XUW#
MNRW][!%O_LB"T%@/5N8[/$6$)S.R=`.'`F;Y<BW7/*/QI1Q3-K@5K2CN497Y
M2)1&O+P:Z+8<,6_^6-:2?&!>YK6+=].40AJ<\TEH4XY.]9$)A9'YJ?4#ROW,
MVMY>.G,_OFV/MEXP8AUT#:>I_;A=;FJ4I6'7`.T#>1\S@KPG@@QG#HZ8S/-Z
MTXP7_]_%9>HHN0>WB-J)B%I81FU\VM+;TH![ZEJ5&8S<3Z^/M_&H98O'.`N3
M#-C](XMOR#L"-]C&P=X^2^]1Q<6AI0FYFMI/'60V#_9V,CC"WG>>V]G;2&2H
M$0P=3^UG4KP)[CKR5V_H+F5?;-8;5$V-`00FD$87D;,2[SP!HUK%LBE^TTS'
L'VD,?VDWN;SK)I&BPQ\VE.@=9&R;K9U:U8D>D6Q@F'TEU:`YES+/'3'"*-L
;R,AJ_X.UUW#??`T$LXO+]^.+(=N3RG>T(3B#5DU3!C-YX^/H]%@(1@-CJ\U
._'D@:O-:J[L9C02+?*8A@I36X-[3(W68#,KNYH,GWOF\A4M4ZVVO595P`(6
=5^\<^>]5Z->ENL$MDRV1DW#%7<]N9L6LW9:>9;W;JK&TM50,TCQ:ZKLTZCZ
MZ10$]3>=D[*H'UJ-)LSM9DUGC\HYEMI97V0=S:RJ5N/A&5=K\WG]F5AQJ[=;
M\$=W)D]X50RP>QJ%U!FIB.1*.G&T0ZJ21HTTZ]J"=_@:GJ&E&XH.G<[0(I%M
M.6",BSM#><`;QTGND6;+VR)E?8#]7KMJ)%.L/S%=Y-:]^^MT>:O5.TR67G'1
M,"4O@2%6K&ODH&6T`I'#HOAX0\@56G)=<J@$Q=`>-6#\4&3*6.02@Y$IHY'1
MX8BC;_Q2NT:KM(>Z1N>)6;K&%*JC**R17M'9*8[6)4>-=*F"SCIG9&"8<M"/
M(8Y>,8_)$YCQ#B=M*,E%X_BI]&AT*W!_V?2Y)HBW`GFOE^J8E2C[8;/[1TOF
MI41M1ZL<H^L:V;2"FYF76(Q$(M^TK<>.*=F?JX:&KU!#D7C>4^H:XW-7Z[)'
MHV(:<1;X`(P/.6V-73;)&=)O6XH>78@5E9C^"-!2*N/96M1RQ4*0SX^6E@K-
MG'XBHJ"WFYI5RO:26[#-F&[99TQR+Y.8,?&D64[,:D$.O.+?.UX>>H$/L<4D
MS;]U>WEY\\&#P^CZP>\M4QT@"RX&-&%`$P:^[`FITBO55=9_S,V?,X8Q;?TW
MM#JR_A,-K:KUGU<A<_WWG?T#/DS=S-PSET7MQ]QV`\M=2L:^L4*=NCVY6PV;
MO8Q54UHVM1F3(ZLO.^%*0C]/^Y_U&8"I[7]Y>/\';__+JOV_"KENZ:=Q1GIO
M8^\.W1&GS03U9C4G-L+R^:,^I>'_5(["[!M`T1_\,[^9ZI=05WW^9]9K/VE*
M^P\M1X;O_X:7HRNJ_;\*F==_FLS8[_WBMWG`.6VTY`N>K@5#HB`71AR;#U%8
MCB-:M.CBTN9(N@R]_78XN#`:OO60@SW\2"*<<+%J/#QABVPD,6QUZ)$"RVK8
M)57&?OWAN(96AZPZ]NK;XQH.2ZM.Y\[MK9;[6\>YX*W1:#@VN=I2MPJW#L=#
M&W\MU_'-J$L2G;M^;<97@FXQWS](;"8?R,A&;KG86]_92&[X^+^9Y,8"=^<S
_W[[;18*LP4^S[<,#<6=EDS9D"RCP=.AL#82F\++ZZ_;'85<C,K=\);1X.ER
[EBC[FB,TTQXQ&G8&N/:=@;(EO-+,L:]TOY?XZ;PC&%,[O^CD65^;FC\%PZJ
^=\KD=G_F[<A;=<`VS'L'$R8=YNWM4=:16P$-)P;KM</#Q+;O,VDWJ&FLBY7
Y,+^",N76_J(:WE#W&A8<)W2"NVFQG;D!DQW/V'#SRWIYQ[56G9?+-^[>@@9
'J+2PU[*W5W0<!>4[N*-1J5<$'N;O48^(()\['M<+K6;<N.S>6?-+^^[XW;S
@FLN&?>B=U.R<PU1;QA<D$'NUFM^70;!,VVLW^3!]RV_(?C=-^ZDT]F#>KM%
]WS$79/(6$.;29NA\+`A.CNCH43<EIH(#"7$LT_<4#Q?;[;$39E4@C8VL*:T
.M;<02(5%1=M,A>%N::FAT1=8C^(WEG^T5C/J9VX%9=5>$[MK)M[7:G,BF6=
;AF--7)O/6$9N;5@5C6V3KMFY?VF\3%(;EJ>;R\XZG;2%HE-K54X&9\)1[8\
#5$5X58.C+K6*1=;)U016\UZ12PQ5^H=;;0)&-;21TF;M;"TEF[F&CH[VDS:
```

```
M\T:?5MCI(UO>A").4PE74V-M&;=IF4]LPG-M)>_;Q@!NCD1MF>9(%.A42[1;
MPVCK]ZP[!1/:>YCYL(/#O;T;VSON[[#A!B_RC25$7:)M.CR4(MT=+&EB1WXS
M5]/EHQ5CC8H](\Z>0!I-::VEH]P9KVV/<I5R4<9[[Y'6;):+H_76,+>?ME>Y
ML&5NOUEOR?TZ:8HL.\I5'HZULKFS-]P9P(J]8QG7!1A6D@XK41<KU,]-LQ)W
M6%FVK,3ULUKAI%FOU=LZRYG]D]AWQ<Q];=[A(>A443\U)4[W-^TU867!]'B_
MW&RUD3U6%,::2=K-K(Z:H?R9P4PJP89Z.J>9E'S.1W3<2XYL$]WZ^&JP9H_?
M;<OP9KU)3Z7DF_5<L9#3QUM82QUF=A-D(FAT@51T:SE>BWGKTT_X(#67+U?*
MK3/6;I2:N:+FVZTS[?B8U]6%B6:3N[N)`T?G.M5RLE:CAW$F6=T[3$NKX9FM
M[K5;TZQN'FYO,_-J.)/5S7:E,M[HACG]L5\/R.B&=IQK5UJLD"N<:*Z7.,,&
M73.&;$2D#==KRN4;DMA_<TS/+A5R;9WZQ[;>T&HZ=62\=]1.^>5T8M](EZ*A
M&$:=,4Q\R3%,)S>"0S%<ML<PN<%*O#-IL*!YR1D=1]N,A8:,K;@:"\UF+#QD
M;-756'@V8Y$A8[=<C45F,98R*J]I[+;=V-`X;U*DUO<==L)!RPY+5AL5K:K5
M6I.OO^GX^H$S-N&0/3;Q]BEOF+3O=1T#M5E2>#24PG#8;E,\4,TO4NU:^;C,
MK\BBK8I]42T^+*T5M"O69/NH+7]&(P@*A0\BQL=T?[@LPI%7'5,^OMP7F_)X
MBUNOT_ZU6F%2C`^'8^SH$T9B.!S8U6)LQO"0<D).*9EO?^]P(3`^JNGMM:&H
M.CH'.CTNHN.-RE&HW:BMDV"/<,F7;P0P)A<3S&T-=[%A1S=1TO@ED]LS]ZKK
M$]*[\=ZP+4<OP2]:[]V/VY,YWM:]]8UA6^@D#C3Q4&UQ?#0.C"PR![#V?H%?
M:HLZE:0Q.MJ9,:L.C$9MV94]Q;0)G6VC?X667B9DH9EJ*PS9<VS(R:Z.R458
MMDQ1V<>/OT>-1=R-V:?5PF;`6BZ18\?"4.<GITWFSO5Q"R7&QG8^OPM&[3&)
M8BZ<2(6F>%NV>UN>U5LH9/<60D%-][;B\+8RJ[=;#F^W9O06#CLJ>-CTQGR)
M9(J>^,XUJX]N!99'!X)..Q&'G<BLP3MB'9XYUK==FJ7P-L&?N4@@G]/SC:TC
M8G[_0W32XVK$=$>AT"R.5F9Q=&L&1^'P+(XBLSB:*;C;Y@/A1B^6MM8:1ALK
M5CF,9T3<6ZOY!`D/(.)H/I$)S6?DN1/F<S=HK<K,$M6),;79G;WJSY2ZL;Y&
M4^>,Q/ONX5B)3N7X-8M68?C$F8]Q^:B43[^-*X=[[TD;9S?BR<W,)NUBML5X
M1?:"SE4&TS;OU%LT`M)'UW<LD\DADZLPZ5AQN)S)^)#)6S")!>,K6MT8LGH;
M5C?$J^JL88FTXV[(L6/96(N0HT;KKD@GQT>'N8=:#2_@B(L[9=,,1L+V2]I4
M@Y&PNT'LFY:3++.*R1LL<NQBV4V);'-=W75LP1XU%1YG:G*L0BZF0N-,N5P\
M;*:"+J:"XTP%W4T9>\D=(QNLK:?VQX0/3T'#4]#A*3@\W#'F>@?6<"=D#7="
DX<[H<S0^!/A[.P<CK_M8'A='QKU2J\;YFAONH54W#GF0R+IV106IPWD-"EF
MZR<:_YV8;BSH,FSC0=C]X[E5E_*R["0V[';0-Z02Z<3NAC$>G>`]L9=R&<]9
M529!+S@IZVR]=<J+L%:88"KIS&"LFR6'1\#3\WE#[K`86IJ*%T3?QB>UMNF.
M?.)X4O[4Q,V=X9'G;EV\:;5!:W&3/=M7@#'^M'E.3/!\]&#7D0PYGCMJEGD;
M;&C-:ED7RV':`UH+T]@N/6PVP5PZM?X@/;(D(V9>R!IRL'NX(WH#NI&HZ>,'
MG+SDDXZ2QR#95O+BT1A;T<]@,[4?WQT=-=!R"3\QL18Z2AP+#XE:L9RKU2AI
M8LV0;CGSVB@F:#F19&UT4FDSFG`V^.6A!&JF^0E&#M<=MQ*P)"!R?71!1-S0
M.[W<.@CU0>ML_7[\<%'\D;P?7U_$(?Y'KE:\M+WD.N-6#IWK,^)2.:VI;*=V
-H;G+;MUF=SM>J[(=MJ55KG!.R9QP[I5;UYRG4J\+0`FIE<G'IWX7F)X1O32
(Q)OU:M4L\Z$L;UF4:,]O9>VARZ;7HXE1KS.IW+'IW+L;=ZA*]?[8T[CLC3^
='#B:7G]&'<:UX=QIY.3HX:>?-QI],T33B<FG$:'.#;FR8DQ1U<U[C3:_?BH
B:8R[C0JKGS%WLC4;&0,9+R=PF],S\:,@XPGS=/!U'N9U+WDIK@>X(I/SUO7
CX_I#=KI]-I!,.,V9#0L)/8W@J:%57.7@AG%EJC*G5SEH;$;9`:CR8.M7<.J
#P,1<6?0N&<F;\3E6JUF.4^7O.DF]^PF<7M/W!:\NLG4/3/EW*2\ZJ7L-PHM
8U0NTPVFM^P&Y:7+],5*/%?;//OH_4GC381L!8HARG"!AB9&(AXRX\!\N):G
M(I&;Z^S(H/T"%,LGDIN3*H@ID4?KN,S5)#)D:0*8ECUX2(_M8),-KGG,+DR
6P69;#)US\I+'RZ"4RK(9(/I+9M!+)E;Q3!3!4GNIVPFC)$U3V95*]+K.:V=
'W'YKAA1?<;7EI2MMD1DKE%UF%)+UY)6IQ&1=S/2]09;.^,12)9J=>.9XAG:
S%K2JEZ16Y--R=PU.E)IP7S+#.\F*?+CVJ1K#.AL]!VZA21V2SL;+)V86B9D
8468$%NIG1969C816H$)XRZF:4.<<=AP3V9H8C)#5@@A*X30F!#&V3"S*CQL
8\:$AJR\B@R;&,TL[TQUFU\LQS82UVC0V>B6H]`C8;%G<I'QXV/]K`A/9@Y&
5N!G99*G4-J19=$@//'CXR-G^)%Y%#4C-\%/:"4M(R<[TVC4"&=E@J?P,OGB
GN2,*'H+GL3Q\=';1_1D=[%L1F]_S;5NIN[5)I1&ZEYF=V^7(WY@3/TPD!'[
?&5WRVN$CU^N%QGODD?OU,",V%AD&)(Y;ET$4I<P)/8]V0Q%[",85T-NR:8+
TJ2$T_G,KO60A3WAS:IY@WD1":#<&+_CR#1XM'80/[*6IR88%.L/_C7C-2NS
3V_H(Q[^.'T(@-JD-&/^7)\ABNF#N)7F\-0HID^:]7;IY.>*)9^[7B&B:_:(
1G[A>;DX,9HN52PYI8HEQU:QX63(NCV]CB7'UC%WBU]&)4N.K623XOAEU++D
V%HV+3>=`5\RTI>L97P&,*F2\=.9Q&Y\;3MA)L*H9.[SA+%6-I(ITXRM8DV>
;;CVO?%)HR%^4@ROC!RQ1]@8=@7$(%AZEB_X'CM)@KF0S5S(9BXTP9QQ1\0V
P<-;YK&Y2+Y8W7CM(Q\)Y\]D')>$Z8`8!Y]HACOQ:MZRSL+,MQ+UIT]3[RUP
'RV,CEV'9'RZ&=VR:H<QI7`S^OLT2A+6QEE:#H5-6SXLL[I;NCW%4BA\RWRR
T8=ME:Z6HJNPA%4VZRU]XV8<ME6V*:(E'1^]`FJ<@[??GC@U6L"RT&5MV"=E
E@WWD>4X&_9YHKL-:T`_-BVVZ:J[C:'!DHL-^QS:W<;0L,7%AGUJ/R8_IMI(
3K4QU(^YV+`OA;C;&.H+W,HV-*ULAQJ26[G8%HG&E6UP:MD&7T+93K,Q6]D&
7T+93K8Q4]FZM-M+EDO07BYBS1<]/#9`QLWUHF2M6&[B'68'UL8<\:I/\:)*
E_Y:G(RGTP>9C<3]G83QJ%00%Z\-[5&Y8%PY)GBF9]?WCK#A(^*<<NS81[6+
SI&1?-:JK-5&GT49,BY&G#05G6B<!IOCAT?#\<64<W6&^(KQT<R1/4KOILER
?@;++>1$C4=]]@#69`#',P2OIZ@[K1M5R%&\ULJCSFM-AUY].JZ6D-^MI=H6
\_O];"O'!Q=T3XL6@'DXU@%R=K!4.Q#.#C1_W;SY=8,EEFH)<3R1:U;.<.,Z
7GA8JW<J6K$D'@&87F,SNUN9W0/C625'I?77MFH'/(RI==<R(V?FP>B(F=FM
""/"RJTA*[,:,6P((WF'$9L-HP0=0WI;"4[).JI$R2-1A8*1A=$:-/O@WUS#
M<-;F66MRDK=KY/G(//@*D;A,^T\>K<D,6'5K0Y><_MCS('^9OB)I]!5&A;EZ
M-,:5Q65Z%A&=@R-[Y1OJ6PZ.+A>IEQ0M='C!PLO.)9?N<8:X((N.?S%9Y-9E
MCXV4==?$>K99'!_OQ;H[<LOA9?26BNDE;-XG77%X&=V#:'J)V&_&V;R,[H`T
MO40-+Y&PP\OHAD[3R[+A)>I,_O)X+RNF%V?R5\9[636\+#N3O^K%)-'QUG57
M,[//$87D8-$W=J#V_I0KXH)IA3D&EB/U9F%:3-[_.>."J)A[X]VC$YHI.E>+
MB]EL3>-&7-QB$IXQ8R;$Q1BRCHW,[KIE?5)<(B\I+J+_G)@U!T<+4^(2?5EQ
M24^,2WJ6N"R;<5EP?264X>&K]UJHJ[S_:?1S/Y/?!#KM^Z_+H>'W?RY'0^K[
M;Z]$UOL_CYN:1I]`I:_7S_!Y)_<7A]J_W^YXN[G\=/E;#@?%=K4AOFU>I,<P
M,I5Z::*K,>?'?0K`<"O?5BJ#$-^7='Q%7;XFGS[T[*5G<V4",V:*1=3-+\>+
M?1BV+\GKX@$<PQ3YY^;H0^3R\]1ZI]PJG#"?\+;`?FI><,3'58.G!7H38#08
MND.7;[VA%<KT'5SYWG[A1VYDI<^.F-_"R(V\HX#W-#X9$7;W+CTE\-.1T9[Q
MJM8W?G#S1TYS6I'I;;&]F+[E>_;=']];>P)>X[>+1<Y3"%"=6<;HX'%M<0VX_
M8%J%Y]*DQ'SW1RYY(U_=\@H2(HO:'QJ*MN.7J&]O.<M=?`UFXW!'OB+2?/GD
M'8?'Z5&8;GJ\T<N8,[Y;XS3EJ%%F`="J"R\1E\QW]@6SAKUVD(B_,Y0&6SN>
M;*8HWV$R+MZ\\K1KM+Y38_9WV9@MWR41SA+_P-9_$'XY/Q9_E>M_NU6^U`?@
MI[W_-Q*.#+__=V4UK*[_KT+NGWLW/S16+8@/J=N^,X8CLEE9W[%"6["^$05W
M]N_7&)]E'_VR^ZP?=+>>A(WK#YFOD>/75ZVX(#[-3D^DB"Z`_B@5\VX?J1[Y
M-K7SD]2C`<.9[0G<,KT1AE%SH7VWUJ?NC?O4=$9\92O7%$_SE/E_QDMD^-6V
M7!.1%;>U]^5GV1V?5Z_1![?I+_H(.K_XBP>`;%]M9[X'H06Q6]3QD7N?W$1/
MD_\"CW!+8SGS,^XC7^HF#Z-?Z::C4[[%;N13:%P^"2,+<A$"'TVCU(FO:]F_
M+#]2)8;BY#@\:^4P`O^RF]0OE:[2_SMG"M/#F/;]AT@X/-3_1]7\[Q7)?/_O
YD$BD4K'=S>2NUN.EP`/G[!F?F_K+5X%6H&3[]F.5<K5<DMW'N/N<LW2R+%\
O5X9.<B#=1YKU\K\,!US65@9BMQ7;77EJZ^KM']\`6CF,*:V_^!P^X\$5]7W
'UZ)S/:?WMO;3CD:OGD$2RCV+[HX/OIB+K)4M6JA<2;65VZPHMY:9/)/O5E8
Y+,MG5_<Q1"H)7:WD3?GP5:SHM5&;8\V>R-JJKW_O+K2]Y\T/K2M!"K%&<.8
MOZ[&A[Y_EMD57W_Z94HL9L^>-?\QG0JL9Y.[NVFL*(9H+7,TULHI[>H!6JU
8WIMK]BF:[SXS?:&:N-SP8&6=MIB=VS+=QEQY"X+6&LJ-WS"V8)M.85"C&\G
MW9]T>#M%6K\XBBOI8%\N57--1PV:>3?*+4R.&4:-VS=I.B+SZV\Y6[=-%[,
5]R,\V-7-VYW%S*=930S(UY__76>HPD^V:,MT734^.8G3>6D=[U9%U-`1U8:
!QVY:7Q`6>0LVVO0;)//A)OR38=\/HB-Q?0Y:+'?H<5J&JW\YIIG`;O7E*95
Q;M'ZK7*&<MKK$W;LHMML:EJ:+&5NVHTZ[R[X"YUK64W8[WV0[<=7C+_'LD9
F3O#2?M`9H0Y_S5/./R[Y[5K>>N=7(/'C5><3+'<'+$7,7V7BY/=C`_B)SP5
F08UCC&Q]=H+C+9-=5A`ONJ#5X:\KHM/UU8IGD5K\BS;5+$MWKTAGAZ$=Y_L
D&FI`7[*?!*]@R<-><?5.EN`6\/+3KETTF+'Y5-N.=<B]RVQ=M%J-TVS5M+"
.\9**,IOZ88C06:K-ZHJ-6WZ>\%H+90F'*<_U_=V=O9V%XS"%9^X-<[3W]89
UQ9+V<IN\B2*+LD(W?'A[(RCS1H^9(,5OL(WH@_7J.$);\:6-3=3HV8F!!QY
>0%'I)E)(8]DSZ67H*]R_=_)/>1#Q(HV:QC3UG]7ED?&_^&(&O^_$B5WXEL)
=N<NPZ!.JQQ[UP_V4BE\CNLN/O;DKY1K[5-_J=;V>[VT@IG9VT^GR*._FBN<
T(2!WAK,_(5&F[ZQS?LJ?VYYE?GU*KUJW%]ED2WF1T=UW2>"7<">"M-"J5SP
/]*:=,VX&[$.RR_AW&TU>=<W5D.F\`IC7*>D7W^MSKO$QDFY@"1L;:R)%*1X
-+W>]77>FJ[_U)[Z#TJ%@O^6=WUS.[Z5HGL_?'[3OFM/WA%]!L9_Q'OF9H[Y
2Q2&WBI6RGGY%Y\L44O1F?_8OFK&?^XG$SS(U($(U-<I5XJ%7+/(;@0*?!CF
<CBUX/7NK?T>Q2)?K[=6HH$Z=T`N[P12=P/U!?YK7?PJT"^OEY;5S9SFW<CO
!.IWV.\$4M[7N</U!4JI2-8'O,S8];>9G]N+&:X*DUUY#;L4`L5J0;BWY=Q"
I<C\:7.FP*[_ON$UL']O;_?=.RQW7/'R_^\PY"Z_G/@WXIO;F<W#]]Z39XQ0
O%[>J5XV**..^7=RC;O&KP`W9$6AV:YY^?^V@%[_,76&NGA_GQ]5GY\U*_R"
Y9EN,?#_+^/9^,'KG<U0H:+E:E[Q[QWOZ\TJKQR\O&TIN"%B_8MI_U=?_YGU
Z[_3UW]6H\/W_R*AH+K_]TKD_EUOFLQ5-.>2S0TF%V8.XP?IX,8![X=\SO,+
],FTVY@K3ETSPGRF<\(#\MW0V7?OLOD?!N?MVU=NC`EJP2?6A0QG^LV;QE3A
@ZLM1&&ZZSA1QLB+;K7YRF)3#RNSMX6'M]C-FV5'3'TR60L^'AB=NFL[Q`/E
AVQ1G+[H9=M15+/'HR;C86776Y3Z!3,>-2,KL"VA-G5$>)7VSZ?<]$##2_K^
8R@46AYI_Y$5U?Y?B<SUW_VM-#T\Z%@!MAVS=OSO\VN2_#*LG/U9)[;2:]L9
6E!*)E+,9[E;8K?L'XW=3MQ/;-/#FYGD[D;B@?6P**.'LV\O.!V&QCD,#CD,
CW$8#@TYC(QQ&`HO>%W"7MM^Q_$)8^-C$"ZANSIU#]_5J8B!T[$55_E!3>F:
Y_SQL;DI@?82K-B>OEX>^R2V4,@Z'>:_[@3)_TV_38X?([HY\N=-LO`^.Z1U
%?$$J/T#>P&NH1B\S[;K'<-AFO<V[/V7$@?L@9![,;A9U\?.9#5-[M[G$^<-
*TZ^X.C+<J13*BJQ3&HY/75YLXYT;#0!N^/(6,?O[B?LY0K'1@K$BR`T\6K:
*8_2.<+.[#_8M5Z@MGQ[2E0S-M?,MS(V&PSG\7V[\VD90<[-%/K&YP1<[]I?
UK82L;U4<JU2+SP<R0WFIW=2ES0^Q;1_)W%\'J$PZ4O86XE09GOO2&[V3VXX
>P,*/SRE1K@:L1<G$R^<O*01>Q881ER^W3'5B/U5(T8%OZ05>U$;5E8N&Q5[
!3",C*T'$XP<'&42V_>26_>LMC*MR8XU$[3%94)CGF1F;R0VEZXNAAEG;"+V
:\!LAE+W1LOIUF5CPXW\_.4D7G7&#]^S)^D*1O8.AXU</GOEVQV&S40NWPHV
1W,W-'-C.MS?3QB/A*WM."WYEF>M>C8KZWN[Z>36X=YARGP(<-:LL5FQ7R,0
EUD+VV;E<-1*]/)6AHU<S<K^VKVX<T"W;'W@1EX]L"6<'5=R)5W,J'`[2WQ5
0>=SI@:&4WGK:D,_Z8+#W1:U4Q85S[G3,IP8-\7WZ2_FHU9<J^-5[8OB#<?-
#FW*9+[#I?V%![O&RYR*KA>GM>V]]7<R&XG-C%P,F_!0IOTMS&,&*.,D'X6C
#Z#;'F*;];KX_OB0)W?7/"PV^BFB2UQY+A.RH]%?)61'7RI##LV>7<Z^@CS/
/H0)CPQA:!NMK%'VNZFBXM:L+\137:/[AE0%RS6WN_*SM*7PR[CLAU_&9=]I
9'=O-^$P,NME/SQTH74VA9DO^TXS1Z-F9NM\AV)S-&)FULM^^&5<]L,OX[(?
M?AF7_?#+N>R'7])E/_QR+OOAEW+9#[^4RWYXRJ5VUL)V6G'6&9_(WI%M>M;Z
MD=JH]Q755=9_#Q+QC9U$H/J2]O]%ED,CZ[_!Y8A:_WT5>I/=,TN;OB6OL1TJ
M;;9IE+;7N\,'R?0^&;,"X!$CL;V,UYT*/6Z4RE7U=JTT+Q]GLFI0P.M-T]>8
MQ8ZF5J?.JO6B)KYQ1+O&Q.CD+5:OR<\+Z(Z'JVA,8YS!4U5>L1DJOKD=H&>M
M=&DU7^?'BEJ#=S_T3;*Z?,**R8?#=);C@R?:!&<\?D4;G7+-5KE0T;P4!/T6
M.]UXYW4LOAN-YZ[%CB@SS3RV](AVJ_Q(O):33F:M9Y>SWN-V#5\C]B:/1;P[
M99W"YYDD'BK-%5IR+Y?CB:AZLUPJUWCNBCC3X7J[9297;S<:]69K43BUY8YX
MT%9NE##]M&P/GSF^AE3(U5A>\](N/OHB[YMO&@^Q&1:6V$:[6CTS0_5FLUG]
MQ'N=57,/>5[1`WN!@!B,)D[E][N,,$O%/(UDZ1571<,]W50G]]P(!?:FPY/3
MAS,@NJ%N>MRK\?*OTYO;&+T=G[+(<LZ,32GTF0IC9PHW;.UA8?X?>^G]P909
M^EDU7Z_@JU"6$QY4D1<33YB/^UU@O!LH:?0$?[7.(WLG%(Y$N0GQ0SQT7");
M;9W^E2?EIEGF6Z`/LV%/1NI.:-D;6K:NC/)9NM,(/4OGV-R%8&5JJ5!XK39*
M9+@$CBM6QASR6G_CAI6,&S=DQN)92%&/EL@6K_?T&3%>/4?J?<`*@#9<I-Y)
M[F?6]P\W#Q+?OQL2AY*9#3[>R*S'#Q*9^!H-RW:2J10]9+-^$$_=2Z2X.SFO
M"BSQZ(G+!_-___O,7Z:P_'5&U6.BS/>I\)B?^%MU/^V>^-Z24;1Z_;A5K;:7
MW/94V+T/[?F9'.9HZ&QDQ])EO<L-3I>3/?*T,>KJWK&CRE;S+Q>ZZ\ZK2WN?
M:8?6>.]NF[5F\&YNYYH:IJOW(I]$%TYX^[B:=U[M,[2WB\5BLF5^V9?R*\EM
M_(<KY]++"D/L\5E='KO_A\LY_@LM!X/+'K;\LB(P27_.QW^3RI\Z]G!@>47T
MS/YH(!0(^FDX<MF*<>GR#P>7PU%5_J]"5RI_/OPO5K69)X#3YG_!2)3*/Q2-
M1*/!U3"]_R,24OM_7HG>9/N\1&E02V-&/E>AV9S>$L-(KW>7CWYUHS[0S*IA
M<_R(ZD:>_(BYRR-10?C0DAX/8#JM3(NI#\_?2H5F$>8<K5#FQIK:C]OEIGR0
M3./C`'J.Z`_X_&J1!JX5>ELXC5J]?-1:;K)BO="F%VCD,,.B\;*,(08M=Y@(
MW>N--QI\*G?C!D74;U9@FMH%1#+X<-EX!0=[M]X6DZ.V&%+GV^5*,:"3BU;=
M*W_1YX\U^8A,O=$J5XW'GN@&$O/O1>2C4=P@GV$^*M/[@V168&)%9\TIJWPY
MB4ZO^FB4\;2<&):7G;Z<\?+OA44>\A;G+1^S,WZ*SUKF^50RQ^>Z/(/)D9@=
M%IKE1HOFQ67QTA$J13E%%]-7FD3+B1+WT/3*:3M%@>;C/$?C%;U.SX09J36F
MI3SB2R)V/-IY?D3X,<M.3K3E[)H2D_.*UY@=\R$9I5AOYX_KE:+6#+`C3<RL
M'E&4*`!>ZH5ZXTS,S^I5S3)YPE/*(RAWS<LWR/,"19G1`>;+/<KQ#*3-,8_*
M.9%X4;0+`9:4.01C?`9YYJ5X%.JUXW*I+9<;:)-EK<2-D\NJ>`[*R,N'FM:0
M\V@1.KT72F9ACK:/>MLU/J'B\:%*JYWD>($W=9YU-V`_<,+_I"K7.FMH.O\E
M9EE\CE4JMUBA4A?#[-:=I27^;T!TJ:CR@7JS).8Y`7Z&NR\4&?TR?-(WTFDV
M)1L8CN9$/1]7R[FC?$X_84:=E@/4=9[A(GF\P.A[P#SO1:92N39%'F/>6-2H
M-N5%.BOU3L"<H'KYC+A1U\NM>O/L#CMIM1JZ3,]).Q_@-62I5*^7*AI=.:R6
M*?L)KY>J=4V3&5HPHB);@U&5$!.Y8*3S1DEI$UG5:N8*&F^:](UHFG63WT>\
M3Z8V)V>'SE+615I0$71ZR=V)2)FX"4LIYP:\?,[+CLD-[Y/.>`O*T9G`:+%-
M3J@H,/F'L[QDNN4DGEPUK+DNI?9[2Z>W5C+FHS74+N5L5QYFSK23G8=4_7FG
M*&P9[]+1E\0RB;[$)W]+>I5>A*.=:@4Q%Q1N'3-T-C1#7VKKS26>[>*:+A9.
M<K_<$QJE2^E*XS]GMV,[:_5`]C`FC_]6EL/!E:'QWTI8C?]>C:C$F-\O+BE+
MM->@LM0JE)8*C;9?]"(%EG<][)6[858*MY8+$4T+:8%`(;J:O[42B2ZOL!`O
MUFC42Q_4<+?JO7GSYCC+L1CS1VXMKK";_-];U!LQZRD5WM?SZD;.Q;,JHV>:
M6J.2.Q/GO#>MLV9G2@&5:Q5R<--+-RO]Y4*]S:__\MUCYKO:Y%!3/&CK971%
MISN:<E6=I?C`8)TV<^CLIR*ZH=L1BF_H]K*(L'@Q+!\#\7!H>8NG+R-7ES-M
M?CTI\=Z[G6GE13?MX]>@5(ONH-S@1Q?M7X*3VT5NE%MYZP$/H9S.+Q4M7Z'@
M_QZ]2K51&'X[J77&)XQ6<GJ+!^C_GM/I!]Z;`L:;5F\:)V@%6#RHQB.72>TF
M]_<3Z3`\X@6G]O=>\F.4";SM4B9$0Z'%U=%,&$T8SX%C7HN&<D":IY?:MO+T
M0MO=P^UMQSMMJ_12!K+@>/\G.>;_9$I:+5/@0UJ9\$9AD17T#!\+:8MRUQ?_
M?9RIYO2'W//$]-K?ZRJ";-=&`N6UAX]K<T5Y$RF]9BRS8SV3!\].:!@F/^E'
M0W`QN.2EP2KU^L-VP_X^!)9KU:OE0H87G.]WJ99^CR?G#ZKT;CO>&__`_B-#
M5C-TQ\G'2_1'BSSA/%K.ILKNS$LI89F^^FM:1TQPA5C?Z.MRO9LK<@$`K=7
M?SMY5!16Q$->:FH/5JJM2L5H^W:35+Y!Q>#O`DL1I:I]+TWEVY0!M.8LRD&
V>%@*,*VQ+"))6N%`(OS*:$XI1LOJ2@&>)ODGK;+!:U&,ZLVCY3,M'B#$FZ<
663W,<4*!X*\IO`S;^#4&[)@Q>`^=R9><R&G2'Q.)!*MG1:TAIBMT-BS4L[1
#3_S?ARL!,C&N[!1S[=HE3@G1ZWX?"$<\E*[(R/-Q#"1CQ([G4X@)Z(K!O85
Z5!?VDZN)W93"7[1#$H?A[4*;>VRSX1I6%\NB.I2R77H$T:Y4A-C9AZ%3K-,
P_9%1G='.KFF1F;XF!9[R>P99L2OK#L<T*<6:^R->(HE4V^PM7@JF5HD(T?)
]+V]PS0[BA\<Q'?3]`3/W@%;W]O=2(HWPK"]31;??9>]D]S=6&1:6=R9X],A
\8UJVDA)62D*D-$+1!Q1,"J^.3>L\`E8FW:IE?B$L%FCX;A]?QN-\;D9\28Y
#.9'TL4#XFWGIE'-6*ZJ-<LT:::!S!EO7@WFYQ.$4ENKG%DS`\I7^I*S\#+Q
`9,1H9#IBR@MK29N3M.MWY:\O\:+;J?,)Y85]EZNHG7TAV7V=J60J[;:QS$Y
5:!9P_>,2)M/7=V+WT_0UI[-Y%;F'C]N/'DU='SX2H89YALR`\S&0,5<*]UA
"`3&[B<.4KP('5>(-\3<!/ZO))X;-UQOD4P1/*:T%M5D7:X&4"O+TP="Y:UR
6H"B-\O4.[S0:5ZDWS$]7C'$*PD5C'=BU:K86M`6\<).7O/&-*]??-RL,Y^U
?B*Z&YKWZ6(:E^.NSGZBR9K**R?O_ZG=U'A=7A@NKL,4E?SVWL'DP(]S-&>-
[Z:2+%\_U42K$3U=H\Z;6S57F]?9JI]7=G:8'`YB,[Z[_FYF;>]!(H5`-K`S
NE6N:F(#`O9<\*I-5U/FJY8KE;*N\994U!=LJTR\1;;%8U]Y/H@J+LH4TEJ2
6!DC,T6M1=LFQ"MQ>8&_)6]$%\0%4K[EGMO)M?FDF";@6G$XLHD'B?5,.KF3
H.X)$L\=RIBG$>,FA4^A''/;/%CQQ3C3+)UPIL$]F)W,P=[AKO%86#@XE#UX
68OHF43B"B?E"CWE7!>;='T[:Y9=HXEGY'0^D^''&#Z0R^C;(MO)G:2U$2^\
S,_30&RRL^4@.3,VVOW^=RWSS*RNKK$5^=%LUVIR?XA<;Q$#`$>LAP,6XR(C
M-XSLV&U7\[S(>7.EK,T;ZVIG!5HUH\J@\6Z=7WOY.$,L)HLWNONH@HI+`3\D
MZHE8O9#M13^I=\1ZCNRBL;HV$JWU^'9F_=WU;7YU,G7+[71F>V]7WE:/CD9:
M;^=UL>ACUG>=!\FCQNMEGEZ[+/(H1\NS#6H+?-0P')'T#NV)&"Y!(ZB=W&FY
MVJZRFAEDNU;F`<H60)?+0C-G;`YJ\@K9'*GU[R02^YG#W>3W#Q/\8B#3LDP%
MX.9`[,F0#HPHK/&\%>-N*PYXHW2K(Y:A\3ZO'*,=+3S/YWF6UPOS<OOX<&SX
MY6AO?2CGP\LK[@[$YP)$(PU'AS*D2LN.-%!HFH."7)Z/`9B/5X!VA=8;>8QD
MOT_CK$Y]D1\R!CIT*!(6?1H]&4]CA^-*O3/:EF5L=N(/,CN'VRB@T`KB$L_K
M]0KM1ZK2OC9'*8DL,.JQ+W=,7RP5-R^*?]#66]0+CPTK:;^^FF$9Z18[?T3G
MQ%,M0O'+;?JR,,1NMI$^D;>M@MA<)T9#O!D_E,-88[Q)+P-`F>8U;HBW;_D@
M@(Q2*AU??R>SOW<4%J[-*-QEX=^OP42RQN0>JPZO@KK<9F8\)"->-K"Z0)U;
ML4WO*0\OLN@BN[7(D[?(RV&1K?"?]:8P%`K?DB'04%6DR3V?K$A1/JT:E[A<
M0]Q=DD_=T!L6Y$,.M/HJFR)]P*6BB6[&W#?(LRU!-X[D@%R75:0IU_;Y!(E7
MI$CD=RCZ^5R^7"FWQ-"]),<=K%&FN"X:LP%S;E9NHG.BO8JH"1VQ'BV65NE"
M)R?^M//PF)=D<_2*)9,JGO#>B6]OBRH1";N=WTEL)`_%5VYY!KHYV(X?;,G'
M'T++9MM.T,N&_!5:44!L^(2`9X^X,_'_M_?F[6T;R;[P_*U/`3/'-BF1%'=*
M5IRYM$39O*-M*,IQ3N+!`0E0PI@D.`2I)1/?S_[6TMWHQD+1=B;SGG.$YTDL
M`KTOU5755?4CRKN`U@'?F_^^^5S**TCI,A8O5O3A1"[>>JW=VA,U7:C!\]EH
M$QC;^1S'#_U/1/?)HP6_"^D7"!VBY`CA`+X[S&_#O(U0L0+S$V\'6:`-SNVS
M[H_V1?^<4)WW]RG,7+E,)]=2&9'"J0*"O2</>VB"CUXT6*!1W-EQY[U]?G(D
M"MQOBN)F`1U,D!.5`:N%=,B9H`;\07(]Z<5%[6L_4AS*G:*H!V\9G<V7*-]A
MRVEM6:1Z2HS&!8J*)KFM-N4Q%C#/":-9"K$P3R=;P%?.KI<W&272=$?S'"-2
MP7B,-ROL%K?TKI'XN:Z_9!-1.#3Q'F+)<D;:)';Z($,BU=4H85TV^X3"@JM]
M%K$%L'2`OUT($ABGS:.;U>R3B+DBQ$(6)3&REB0-8BVX/ED<"])Q$_!"7?&-
MWIJCO-\[1?IMO_EI((>[$?L,9*N/CA#=BTM)WXWOW;.CZ&OJR8==8/E&MJ.(
M^V7X@&LW3Q>/+.,":YO&C\%9=MP[,3VAJB`/857BGX+DB'%\0!XNIHPVC2LT
M99%D:'IG=JR6?+627@,K\&8$,RJOQ,.E-^>A)P$5*DJMXK([H,$6L2\:*9_U
M@:SMQ<;1]6D)XBWA,O@$A($&-E^Z+T3CF39X1[W#@=ZWJ.03VC',)8L.H-3`
M8@L&/HW5F"P>VMVY`NK5_3#H=\2Z-VM/?([O/8U3#&ECDZ8DK79YYP^=Q2:B
M!.J'2W\D#GF8A`-KCAI&$D'G`6QE/M=S%*,NW,4*<I32N@.!"%DNR`]_C)S%
M`JL$?ICEN.%*6*QCB%<T_(!V3":H7"'^S'731QJFD+HJYC`2&Y*=I9&F=J4/
M-=V]$UL$U8I>2RMU;&&>SFMFF6`!D#F_N,:'E!*&FDS@7=]%RQ:F,\!5(6\7
M,9W3U>B&2KSQK\GF'#AU9@A@D2-U2LKHT9QR1YN5K#GG[_EX)MQ3<D=="AE5
M"*]RXWKCL4<O,*@H2=08^A7.8#::$$<[])*9'S1/F(7HH6`L#5X6!I,I:"6R
MF72;'E#^VM^ZQ\=`!2[L2Q"ENK7T[11+(]<[3[&@VQ&9$R<2VLC?W?BC&[-?
MD^`:VO?)1XXJ23&H)MA=)]V(NTZ2!%68BYT'-H^6/U=&@BCL!)1Y0PNY>'$J
M4W]7,]0%^&IAYY\GU0+4V0_V1;>O3LY]N:"O>B"]C8'+N=',"/+O?DWJ='KV
M`#FY@?WN/V4I\F0\9F\7O"T1&FI<QG-4$JWFN&99)P9K'`Z*T!LE"H?#:7!I
M7UT<=09=(*#4S):V&"].S@>QSZKRRVD0\/#(%8%+#^TB)H'C\NEZ[XU0=8M0
M@=1&.'=*Z.L>RKR%^/;HO'\+K._Y^>!=[TP:MRN9""_/5)>LO%`5&+)7LHL_
MG1V"6#GH]A&QV!R_\]42UQJ;"N%^7WA(*:[1:`*Y)#]P4?4S0U_K1,$@Q5^`
M(/^VWSE49VM-EGPU&PG=FZ"YT@,')VD&)RR,A>_:9_2HHG=W+=7LWNG%29>.
M':EH.X$=J>LYR9,'R9=.V`6)BP\K#4#W<H!N&'O6+[B$2[`=BA%+@,,AI&)+
MQ)C;(WF9>&C7&RTD#A44+4O0\E,)FSZR!*ORS25\>QM0)C5*.)]Y)>!H^0##
M>43Y<H4>7LPR)$I`B?;;2D!Y^-M*`.:O^(TEU'0/EO7KP9\9ZR%KJ55;8J60
M9%A<4W:UE;[8HK56V]]DM:Z;Y\W6^YH2:LUF1@D4ES%S@/026AN4L*X-S6KM
M6^>YHB^5KRNAUOBV$C#F^+>5@"NJ;92P9D7ILY&]6NLUL=9JU4:[L5=O-?9.
M3HII90M5XIK5BF9+]4J#.XG$FU0N,^_:(3=0P<?F4?YT9B7G&B,^^Z-"5`+V
M;W^#'9.]4C;=<]DEM)K->C.U!%ZM64-LEM!:6X+UV*X5(]DT1Y),23<:24O-
M9CMC'/393*Z4K[SDI.=K[QUY"#AWWR,&T[L/F!^'34$F"JR=1.=CPZA[&8`\
M\BUWGE'=7_^HVT]4@;.-E&+;I,(9#3<\5]C3@@P]'%D7?>#YR->6+$96K**.
M,UZP7;O=([M_ICD_(DV+RXITAR%$B+E#ZD0J^^WA(0[42R=\*6\^^79I[BQ"
M*1Z:=M]DU)TFL9[TS!`S>]5]*9]W9[?^(I@1$ZBNIZ0@ANVQ+M^=6KTCFC0T
M9T8=,5T(`O\YC3-PD-;NGKVWWW?ZJK*<;9,/+GSJ'<D+_7-2RM,=L<8H%OD'
MR"HE<EB7#0KC]1R>=(`2FC6)>OC3Z?E1-Z?QZWA?=GYAY!#IQ:?#=]W#OV@Y
M+M"FX'*0ED-\ZIX-M/1'W>-N/[U%_.GXO/^7R_Y[.0"]68GNG7%S.PA*$HJ[
M9#A+4.5,(A):/8!H"U-#3@RQ09!-O.Q%2RSWW7?P,];,[[Y+-%3/8^0R&DL9
M^<85Y>P9!P<0>$&JY6JY^$#<1BC/C1VR>/?(+YY,Y%.NHH\[O1.M'97[L>=5
M770#25;IW:/=.8Y7HJ[3R\X9ZECGEA)U$UL`DMC=?O\\6I36GI36CCSN".XN
M%'O8-6"^E'I6^.\3V6Y%8+ALCR4LF98<H@"MQJ!8E&_$X-G'1S2+VL\=JYH0
MT+3/$9W8ERJ`8W1M$=5C*#=EOQ!=-+ZR[KR70,#N')^]!EC!)JY+6*$L,LTG
MJU!8;J"[@+H3TBI!]<L<)+35/+[<L*'VCYW>0+]SK$0D5%V3R]JT2\4B81?C
MU2<>8:C.L:Z]F;=`YP:^M_?"U512-1GD(?2$O12V<T&N-.A9(:KRY(U&R7%=
MCC5.I,-"F\N0[+08KSC2H)$AAG9-PWI#OB#TZ7I'MZ!(NYOG:_%(5U(EJX9$
M@LZ1G,UFBDG!#5KDA;&^L/I:W-)?^[>LVD^U#WC7.3O4#`24:HHVY:^1.1JY
M5;C2YR1?^QLIM7K_V:7[R4+9.D5UT1!5"1CNA)6"S(P<F(>WX3\$,R@T^7LX
M,V'`QQ)P*Z@O1V-#*"Y$5"S28/(R$#YU=N\,*%:_,^B=\\2S>XQ07ZGZ'#Q_
ML45WS@,MR6!RRQI`3#)!^"[)@L\GWKT\CM!P1W?"0<L#\K[AX>#X%SP<;,=#
%^#D+L4GO(_!ZF@]W/EHA'G9??L^).,M;Q:N$A38&$^Y=UO)[_K9RX&PS9F(
L01PQ`9`6!`"&W5NX5+<`=#HWWFPUWG\02I`U5WJ74KGY"2Z"2/RVA!VKZ*R
```

```
M#G3+]4JK.5GP\G9+N^F]?(=F>I<#O2BG.:PWVQ6I1CK$T/ZP0J+;D>M@B4$G
MR`]JMEP$P$^M0M2K!W,D#RDV?-;A`.0;J40\O>2]H]N#B!2'Y_VN/>B_I335
MM!1OKBY_4G>+\HJG(O^1@ZWIOG#:E6YJC`HU<0D.1S301"`4)32P=D74&<UX
M'KTGQ96=LZ0ML\'2$_N3*[UQ;LF-3RAY4W1LA^?ONWT,-7]^=O)39N/A#$/"
M<4/GY`KG@J(+L>J0U,+HYX74%\V7EJOYA$V`K4X8V08K)\^0C0B*U"6Z1%G7
M+[Y+H;ZEZ0@I^,KYU=G@\I&A-P:7@>&5^1#>05,(HY#F!%E09WD31O<X;J"9
M0GM3O!3!!1WJT94BLSY(-EY-^"(UF-UZ,Q_JG3!5F`+U8MLA.?T\O2-EP+.,
MC&(=*^<"9<^I0\N97..%_<TT92AZ;\]P]1[WSHYH+"([MV<QBUC,FS!W-]T[
MA(6Z^7)3T_>]<64X;+E.NY)I^AXK6#>`!WDXPP"^^60`_V0`_P<:P.-M9DEG
M/9@,L?@8HRA?81/_N)D\S&AGYBY@U[U=P'Z%40NM[V%&Q=^Z;;RR[]">S6SJ
MV90-9"/K!<HGZ,9%ETK$Q:>T0&PRIG8D-A'C"\?\5(71`G:9)?K0FSO(@DX>
M2OHR"H,5L,E"R^#252Q9RM;*U4JY4K9Z:.+-Y)&MI-$5>UF2AK?(>]))]KX@:
MYZ!]&A^50>2#ABQA*5P^3)+D7UQU+S'6P@C-HL2N7'CHE2!%!6CZ,I2?T"%:
MEV7H[C4:BTRW\-!",ZX9QLDFWI-`[!QRC$Y;0H@C=<"LS@PVB$-79FA<&=Q1
M?`DQ*I:PH&;Q8KY:S`.@$7RDC@2?S:<_^4KA_3&27304!O&Z/"I'OA3",^+[
M\"'<#6^FY9L?XOX2TD_I48<'TK>][_1[YU>75N?J0^^DU^G_9%T.KHZ/'].G
M27U9QPIG_GP.#(VYF-C=S0XG/!`^(BE/1?B"!<ZV-"/'XY3Z?^?Q$"#;XI"9
M&5M(TGD+?"FJ#(>HAI.QGMG:3!E"P*IEN=MGK:),1=0>EB(:O`1W,^)"O7NT
M$PD"/$QP.V#$@]`((DT6N#!Y&*%!VOK-$)(6%Y7GDN8?#XCD=7":>QUR)/\4
MREV<65LP\_8RG.37^NUQIL\,'49QAJ7=$UHSB*&7)@66M![48R*R60Q+M:R#
M#I!?$:-NDW$9&YT&8O>N0K2:T)0=I,(MTNE*M$8&CUR2>I)-4;UK1-]-;%U_
M]G=AJND[[!T2EF\V&;.:,6;^..^S3R>Z2>((DN6A37TK1.EX="FX?5X.GGP;
M=0C]):.OG_6)`0Y@Z-E`+61]V5/P(ZE7B)>6IH.D<3`41\RU3H"OR,V(!=46
M64ZPF&%(A%.JF4FU2JX6,F*G1LCB0S>X/$%U45Y3'96LJK%*D*8!SW$+YPI+
M[G(:)%<J?%<54AL#L^%HP%9U[/ER<2`UQ/=H&H6S*;6QL*3F;+>-AP\'N7`]
MDQAXDS$:6)1'&AMLN`C'YK-H_?R#U3TY%M:E],[*\THM6-]_W-IQAGY65FQE
M5E:L/3IO::7@!_*A+3+0,R.3TYZ0^X:>9%;@URGC`;Y063W4U>H9+74I@VY@
DI28KC1H!1'M-N36UTZ-7,PVE0)S8V#LJ:6^N+7GOBNGKA?;F*2@RZI&EH.[
V5Y,0^NUTE+(\H[%88ZWB1-'&:D;Y1$\8;0A\2?RZ_&OVL:4[M';6CKA4ZV2
1UM4+8/"05JMJ`4E"JM*I4N%M";H]%A?FD7+_+6"_*V&O8S^*L@!.4(9E:DN
*L+%_8U#KEIR;T2'T@WL[0GY^LD1(V=[;`NT@?W3C3TR'QTDWHE3@S[(]O`)
8KX2IPF\_"Q;>XDF=3#P=PBXBU,8:ILSU7W]AAR[;?;EUH?TL?&JU^1XX5\X
7E#%E_K+;R>C!42SN8[-(2:G<SBXZIQ89*%TVCT;H,[Q;#V+HV_:U9QNON",
D]RH@%(A.V?BZ>B(W&#YX\SBW#"-]8%#6BY@=\&0>;/;O'9G5B@J@K/-VS!*
EC.UISD>`X;0!)[0%KN>(PMP9E5Q#/ES<2!?8_G.,O!E!O4!"UE8/Z`"H4"I
J@AYJCX]6XBW*D,&Y8!Y6%B[(C<=J5HC:2"T1G(W5)/X\T%4OCR9(`4D=99Y
SE"D$`I%JV(VW\SQVF*PU$*I6J#KI'PU2HX'NPABIBNHV7"4^/,[OD`H(L=J
W&&SZ$,W8FKNPD#ZR"K.%_DL9&:O\38+EA;%3I-+1[98SIK>\I\K']4XQ\8O
L3!.>F\N<P5M0,W3CJ!5HP_R+(.?,#@162U5S:F"P9DO;_"FQ7:62+;S!=2L
P32CB>X=Q3I;D*4Z7ALM_<GN8K0JCYA'[[V<BF%!1A)E+/+;H>!YE%6XV(X<
8*N$>DXX`&MB@B4=)E<++I;KEZ'?G`DIF?ZLAA,:8(M25)NI4Y]Y&&,7:R0G
M(F&Y#3Z1`@7512R4\/!E[?&40XQB?*@I2#W*E]/YSXV/VFY]ID]X040>.5"C
M/T"-O[:6:&W=>2_176@"ZXFBPJ6L-5RG8OA(D)Y\PM"#(?`#>A$X+YK7;(PG
MB98H-A254U[>O-`L8G^*5J.`B+V-6.MC?`E3,_@C7XBZIY2$11"=ER]!,J.;
MS&#APE&)T)JR!8(7KVH+'(K"HPY9(L'WT%XBU&%GN<)[1WOL_ES[:&SUFR"8
MPR<Y7+"R_QS;B;C:M7[&^TCTHV;0CVZ(1Z4?WJ`2$0YZC%G-5U3$]<$$7"^<
MH2X)J.MDC!WX4LBQ1)11_.0`_=CXRL>B*(04\30'6!Y+`7$:,O?G7A[S%:S?
M?K/<U;Q&OWZN0BF<`[AI()7<A[HDKZ-)$'HR9=0Q-;(P<VH;R:JBCU&!#9,&
M/U-IM$FC\3K4F>&R=8CU&W?ON,UA7:+=N:=WTHJQPH(Z\B?N1C1QA<PO=!<?
M^RJ&.WIMK&5%$*G]%[1X](9Q&6*,C5%(WS8OU-C$5U;36%F'P02O\HVE(UA7
MU)H"P17+P_5%D`1J"=-57`YZ(PT6F2((:9U65;[U6#O`0BP97_!^8J.@>,^I
MCU`H;FRM4R_D'JSH2ZZE+Z&,D9$9S6%I%_2#*2+EJ-G3A%WI#HI77$C:Z6H7
M./(E,IVTK>(JB<UE#O1_Q-!,<0)OVWQ"6E%*O$O#I!&A.^<`JZPM%P4!BZ6?
MQ47AU2%.Q6F`@93\3Q3BQ2&]@UB:)QA14<2%#$TBG5XR4@/YY?LX8P#?S#-(
M<(>Q@^@$Q("0W;1)?^I,[IP'O!Z?T\(8^_=PQ-'5-<J%>(5A==`Z!WJ#IXPC
MVC["`W8Q*UL7-]Y=V7JS6AJJ*\=UR:TVE$%3*&Z9YY:M$SH>@&9>3SQYL%&4
M3#3M0`H+RQ9Y$O;W^,?*"?W2:N:C%C(:(CD(R/2HH?H!%]G?HA???V_M\3J5
M;UYH/&U)\;TH8RM=MG+$#GDUF;+WD*Z_R4$=PT&,%LX4P1)9X2EZ([HN^""\
M[@\US9ZP]U0#;(4<=LL9+0*\9%K-LE;":X,]3SFC%;<IL_Q-K=^/.SLT$/(W
MC)LV;-6#^%[4;APD0X4:2E2*BPBWWLAW95@,/V)76!?.&G:-U*$3':GY_1$I
M&UE!C-1)Z-8"NN5#GUH?J-*/K`WUT4-^CLYB1:EH7L;9J,":^HN%N+E23O=%
M@WU7ZE]@>LGEGB]$"-V%=,%9K&&6<H$BMID"_;`8$^2+"2E>HS>&RF`9YR*C
M<S$VR\OR7*!(O%9JA;+0)>"4#L4K1@[%5)%.`1)R,S"A4BN83"&?>T"[@8U#
M:Y5@G#?;6B!:GO$I06\^Z^M)4$,A=/NNN@M")V#);96M2Q\9=Q7P!.^,6.."
JAB^0I1A8LG/Z<'ZA-9>PK27!HWD.[QC)(L)XD/YA)5GJ8AQG37K">V3"'J(
M'.G873>+5JH^Y""3\T4G99:ZJ%E3#S+#!@#ZB`/`&F@*8"AN^?1[,7$8QH]P
M8GK'[K?-H<4CI+@)&;;0U"0QYU&FO5!6]R!B]16C!:?S4?EGRV$A"N)H1$F4
M?.4N5A5[F8B9N&&]6C519$2CHMCKSX;8S.S@V)6"9\*:0X0X$]86,J3VAO8;
MK=:XOE_=;^UGARZ4)6J&&_N-I\"%3W8;?Y3=QGK+#5R>'`7,7\JR,7H#^8(C
M1T-9?Q<;C=\SE.'@IXON93*2H7IM7-<O7:C4O*W'EQ-_2"^W=F3873SO]^"X
MMU9[!^;;:@M>KZJMV&O4JENK>NU`#C-U^.KZAIS@%VSP,<.H7,`SSEQB5LAL
M#[6QU^3&HIH$/Y<@<Y`*3AV5C(8@#\=5J\$3!F4^GTQ60E,T14Z60C("\9QY
M*,K3NJ"P6ZB:XYL0:8+1F[&AQ4@87JLF86-9EI`L,B[,ETIM1GP2_N\E?<`6
ML/'B^4RZ10E1J&@V-P^R@R,\!M3E#5V.WL\=&A81&""U+NCD]QA,<U>0TA^(
M-]0\[BD2?HA6EK*+*,S`1FXUS$;YI.B$E03S]S)"JR/^A3WBPU@37G('SX([
MRDWV3(*C)0+I+T:K*1JJC%3\)I)ZL=<RG30O120`+"P"=T`_,M]=WA2M;2`^
MV^9LD$T`4N=/,S3D`%88M7"D6$"R#)N(]Z8S1"O_6+-!8"/+]5GHC#WE0P3_
MBDG#ZQ>YK>+1`]4"3\P%SNB!"A^H;P3BDN57%34P$3109I'[S`KUC2;WF14:
M&TWN,RNDC::]IEJMD*J-*,1I[\R(;=@[R]M.T0:N)0]_%$`BS^.//_,_KRQ\
M63!R=#YDYG`XQY!R1':C&)<MYH/U8^>BVLK;]U"(LI$`"F*!3+)$J1>^*#N(
M/'PH0%WXB81>ZS>+?X%PMU=0AA"%>`WU6KR&>BVCAGI-JZ'6,*K@G\ID@S+H
MK7EA5>XKE>-CY'JR4E)+14I.&V\W+S6\)SDY>7]J7W0N+_5AQP_]//:&7?E`
MUGMNP>]"2L#*S'1J[1G5)")9]'MG;WO'/W&LA[/."1;WW7U*"OR0FEPO#Z-8
MO@&^H-?M0V.HR[;MA%-8][<!RA(3+Y_+6:]>O;)R'"PFQV8BD4TRJY=H,C$S
M6?'Y,YMI1?[9,_Q2).L2+1-POEJVS%QL-I.T<A:G9<+(6:*Y``>K@%TV9(A'
M7F.\WVSLU;Q4AE@5IW'#36*&OWM&Z"#P"9IZ>'S2>7OY.H>@1*7K:W>8L\J[
MTLG2@W:*6R6!UTD,BC>C-Q'&B9;L>OE)S^1.M%^WLQ'E9V5`"=5&KQ-X*5'Y
M<]^+?GRZG6)S"3RU]/<]<Q0UM!5EA0-#D/9:#.30;=1'C;UJVRV7*X[G5!KC
MZMZ>"7*0FIT'-_43#G2MAO'RX?\,<K"EUD_WY-@^O^R\Z<$(P-_TIWWYT^5[
MQ#3P[M&1RUIC[Q.WXC%,<[88]8`TO>A!*6Q/0=)BJ`/@EJAM]3V"8*C5]RO%
M:D6+YT_2//;"AM[8_A38X?P(75SD!3Z^L6<4@`P%>_X-@K.([#\;!Z4?AHM/
M?-^JO</B6+_RVO)NW`6\L866Q3)N31*V;K$7D%^4B._X;ICJ0>XH[U/%%G"U
MJI;YS6PUA5<[.P;2@!#AL5US2&EM>_,YE4Z_=BQ?@P+`AM'GT@]SF]B,UZ^M
MBX%]<MXY*@B4B%JC7BGNPY@VZJUBM?K[C:G1"&SM`MB3%]9%_QQ#<74/C5[I
MB6]1I0H#P<,5K9G4#-%,&9?F5,:!?O"8=3PSEV,A>>TN2DCD_KRFT3:<6#^H
M:78W:;72]8L:H8C'FAV5;%P6Z`4\UNK/CT_1C_W>H(N-SZ)30$[1L3Q.I]1K
M0:?VZJWV:%BM[P&=:E=K36??;56RZ524/4&GHD^$;E(E0H7_Q.%8_N',_5T/
MU<0Q-):Q6R(UM0!B$43K,:L\2_**AR?G9UV[=Q[11/DF/MR5^SUQRO%OO!)D
MV_1`VC439>-C5O:GA4`MV*-6NUC3=R*25:*J;F"+4:B2:0RI*F$)W/(.!()1
MC!+#&575]J+0+:(2&\9F%N21V[>7J$_-YY%B47JIYUX&]DT0+FURM)[DX66M
M4$"8$2Z0;K:$X]]9WUY>8S&OMDJ/5'8=KTW^62M&>1-/9I/JU"2U6_ZI;QP<
M#[SG1MY6U1??6<MKNB;76A%+@)%$M`1UI=M4\WJ6L)'Q6:5%*'44,CE88!0P
M]C[`P,V+T8V/<1PQ@H!Y3<Z;,1\WP7CQ(FF5\=H27[")\.NR][;SIC\HQ'O`
MA@"BHS%+#FVV^&/*9-&U,&;/6AKP#T]#5.+GZ$]@0+]B+:020%G>(\O!VA)"
MN[8^$7=H$0Q7D!QYMM0U'$OSZ@EI[M_VK,-_(Q='OA2:3`1/_S6@X%^!_]QN
MMY[PG_^(YXOG_PNQG_%Y!/^Y46E7-/R_.N(_-UNM)_R_/^(A6%>^12+S`YQJ
M(;TKC.`4'&@2M"40M/3M-DNQXL7\NZ&A!RJ$/,;V1UX;W=P$E/`KQ/'=WM96
M.JW\V[!TZ\S@]'0TF$N%).U<H[97@M%2,J[;$F"]PEX)^5!,:<D11-L,$:]J
M&,"@A:LY>A'A%1T.*`-K"U>G+<L83@K-OI"]S6BSQ.8<QJ$[4YMN-EF"5\N`
M^-)O">.,B&:J-@I3G#N8%S3=INOT&V\RW\)H_\&<H%PH4`E=@Y'CJ+`F8)@&
M-=,JUH[64W(1P!L(BRY+H+<_>BID]2J4ER\<P(9$)Q4Y`A+!OE:C]*^&8?Z"
M54-XO+K*3-=O.<BOMAHEO"2=4LU*A46`O`J%.8&M_#LC%D==BU:2WDGJE0$W
M31C'U,AWN,ON/*(*UB>0^KP)81$KT[U1,/?CESR$$H[;&[I7QGO8K>WM(6S5
MTM0#&D'U12</+%]8G\*?6(^W,5Y`80KN7"--L`9Z=,?D3W$!QZ#3M[=+KC4+
M8%'Z,RQ[S@'MR5XFE9)MJ3N@FP!O9/\K%+X&Y+%Y<OXV5[1RHL`<:H?_2VQF
MO,FC0";4T2VT4"-DDK+""J=8<!02Y.+JN-_]Z^NJB=^<`'"VK%]@@RA@:1RI
M'W9C"XE/<KE`^1ME@\*F0&#)KM-?I,8LC3\B&\:O&:'CXWW):;8?RZ2RA=.Y
M5=VD&C/;U*J_?2QI2C9>??HBW*PV,237_J@D=MKK^N;90F\$6_OU<K'RUF1*
M9!.@H0)/(3N[R#;#D$YSV%./]"F632WT+\M&,C%NLR=L\/]9SQ?S_U]PULDZ
MUO+_P/@#MV_R_W40!9M/_/\?\?R>^-_[M59KW'9'>_\K\;_;C/]=K?_/!0#?
MD1#@D=4IF_7.EJ%-?0KL<!G,-<TH`5FC?XHT^6:')/H6PQ"W!(IX#>_&=AK5
M2K&6`J;^93CBZ5;#]"G-;;Q(QM_T69F118;>VLLY-LI3.GIIQLOVNS:-!:7@
MX7^!%K4O5!TODDCDFH<:A?.HW+^I5"M';SH-C.2!FMU/<WL>3/S1@WWK!UI$
M%GIP99,K3N[G9Q^MB_.3WN%/UOO>^0EY=3][]@R6++&P?0^6Q;-?9CE=F0_%
M4P%,U6VT:573E;]\=S4X.O_QS#[L((#-N_/+@?W7TPN\CAYT3VV,XSS0"V/'
MN5K,ZC?9MT[C\%#T[78ZM>=PAHRR.O3^]-2ZZ)SU#K$G(`0*W!YDP$$41+$>
M/OS+.U6JZIVR9*?^.)AZ6.`=X-W%(@>!`Z]+S)<%WF^S6Q&B1&;E]8H7Z[-;
MH`)B]:[=[]6#6&<B>V[UGD.^."Y+]8,W4J@3O"QTC,)G6NR)HX1\Q.?AK4+T
M53[.,ICZ([P=R+^@?NB8]S_K/VPLU<;@6;"F"A^+,*30K(0-N0S[%$>KW]!H
M9ES;:WKM9KM9S;0B5T7J\?]`'GK,COSDY/#)C/S)C/Q?'/[OUKE>D7.DC(-/
MX,K^\FN,Q_^%1N,1SKUI-1Z]CW-RPN)8AE^7FX&-K>.Q<]]CI/=S'9_7LG*L
MTW*6T&1@CAZ-"[?FV?H66'I<+!1+5CC\"9].#2"*C(0)<`\M,\)O`87X>CP(
M!061CD7/6',.HD1,T#D]#U3'DUZ"&$[=^82^U^A7#*D>?F5[:PP<LF20`SC[
MD^"T>#(?GI^<]]=7/G80=:US=MFSAL&]\/%&DC</R%EB]C*TVF1O?M5+1)?O
MG!W^9+\Y_Z#PN21VNHS2+L/CDHL`@J4;(/)EPP-O-<?@P4.TZ>=[=6<2!A$R
M!2-*XF8G&/`#RA,'?5:(]<E@TXQ0WSOM(IT23S6*6#T0+2:?`@U+D`+@J6+Q
M@]F']&J`"3J_.I-AVFN5V/`8T/*,MZI'[S(@Y>5>UTS9UR'<4\CXN%UQ,AG!
M+49FQ<\2-NU9K>6`^E%(#[JP(D8@GP:\*BLF_HBKSB-TTG9=1I33OM=K8?X%
M=C2N"`%Q!TQ'!,2:QU5*!X,G`*?QM:`"&.0R@AL1P5T6B;9%</?1'M]+^VR?
MG`N<E48RWGZX&H:$7;B,G$-%G'UGB(;<`H)['80N1?6/3V,V[.=JYD.%O`WP
M\!PM@)WCD/\<SCI>@0YV_Z[#?6GJL<WU!(?]SN4[2\9'9\!:&%N2'J,V".1T
M!BNW7(YH[:#6'_FSEP1'^)*1CM-1L\V1KS5;Z0GLWAF-2@HF<`12H5@$9P@<
M@96/@$D=0?R1Z[H+BO!*LCWX2GCWH`%6(/"0,A'J.Q]T9`J)[]&)(RY'(Z0C
M,F*\*+S'H_O:"+@BJZZ>?MJJNA06LH2BQUY3+24:9S$9!E"I(HRPMT8<59$Z
M3V$6\'"1W">BU8LYE5$CJW04Q('G*;5JPFNK]K?9EG+`8B?J.UB"PE?*%20D
MC^6W"TCAW!4Z%=6*5J-H[14)>1"Q`Q']+UA00=7:'M>`C"OU*7V<HD;A.+7E
M.8<!(S`D`BDLT)U80%I/Y%;$\'03C_W_I6,^#%L7[\J9/0]YB2SXFI5<\ZUZ
M_7D4A&%)C/PU,Q_6W,>V%J5LH`0UO%KED`-W@5P)[$T53A'I2,0;8&]R`0>?
MC3%_>=HY8?1.Q#].?C_M'O6N3BU+@+PF$YR@!1FOJ:;:VUV$#RY-"%",6P/B
M@8!WX=BL"X<L!O/?-Y^K\"=`Z3(6+U;T08*,,OZ:J.E"#1Y:*2#I_.0SR/@L
F)5$]RG,)GX7HC`0.@R,)D0%#'_(W'<$LY)8''3_-SC7T.8M:W]?X,W+>+;"
E`%.E:(U]^2)?XO!]:'R=/CZ\Y,C4>#^(_#U\O)?L#[IQ47M:S]2'$JAHJ@'
;ZF%_T1ICUP%";R>%+$9V/4&N:U*;(VS@!G/.8%FHZBIDRT!')91(DUW-,\Q
(A6,QQC@#K<"F5P@\1,A;:U=/#0)M)JECK1)[/1!HD2JJU'"NFSV200V;N*S
P]*9$MXTD<`X;1[=K&9,$0Y4M'02+-&:7I(&L18$GJ\@'3<!+U0TJ*"(*9E'
.<+9(^;RFY\&<K@;L<]`MOH:3+L.Z4+?NV='!HA[\N0CP!82<F0[(O1V`>0B
,'0JE32FS,2I%TQ9.E;]0-@8%%-&F\85FK)(A:J/U2*A4A(UL&)YAOZ^\O:6
$;<5WLMRFL2V5G#W]"^CX#12/NL#F8"^C@.F"R2<$GJ7R?%,Q6CO'0[TOD4E
GS#4WB1:H"0ZL.R2!M&>*![:G8"[ST)&S]A[&J=(8%VL-\D"B!>@R0;DN3CD
81(.&`Z<Y%#A'T7J,D*=#W>Q@ARE5*9%D!\!#YW%`JL$?CA"42$U&9RA:$0B
0<,GS)^Y240L&NGN0`>!!TDJFRVFD:9VI0\U1=LDM@BJ-:&BJ85Y#<NKH(+`
,RC53`I!#K)20"(065RP6L!5(6\7,9T(G$`E(GZ$`+Z285*FR8#;<6![YLZS
YIR_Y^.9<$_)'74I!%4AP<J-JX#>$5]/(MWA&4QAK>31CO!IRI][%F)$46-I
\+(PF$R)?>Z'(J@IY:_]+89TG[J=8FGD>N<I%G0[(G,2RM*1`/5&ORC2I@I=
GUH3[*Z3;L1=)TF"*LS%SB\?>/ES99IO/G25@OCPJ4S]7<T41""ML/SSI&Z`
.OM!0W@#5D(NZ*L>2&]CX')N,"HA2I0C*.3=KTG%3D\"6;W[3UF*/!F/L3T+
@33/OO\8/APU1:LYKEF%00H'1>B-DE#Q@\[@,@9WW](6X\7)^2#V655^*1'M
U8K`I7=X<46^>7RZWGLC5.0B!`BU$<Z=$A!_8(A%WB3\P?NWP/J>GP_>]13:
J9*)\"HY@E7-"U6!(7LEN_C3V2&[/+_OG,3&[YPAK5Q_0:L`F%\/*<4U8MTA
E^0'+NI_9L!6)`L&*?X"!/FW_<ZA.EMKLF0-EHII+D6)$@$:T#4QI)C%9_2H
HG6D*XHU3<>.U+:=^.%25W;BT!/YT@F[0J?*!GW>4_CB&Z&#)R"?-<SG;T:B
KWP[EOVWE]#Z5B3Z>NU;2V@UOK6$J@YQ_G4EU'33Q/7K(89]G;74JBT-VWL3
9.XU^.*U_6_$LM]PO:\IH=9L;H!EGQP@O836!B6L:T.S6OO6>:[H2^7K2J@U
OJV$1F6_]6TEX(IJ&R6L65%)C/.TU5JOB;6F\-/W,!;\(_CIZ:M58+BW3`SW
F7?M;(;A3CMF_YNP[#?=<]DE()9]<RV6?29%T$IHK2W!>FS7BI%LFB/)2$2;
C*2E9K.=,0[Z;"97RE?>=-+SM9>//`2<N^\1@^G=!\R/PZ8P`77=0$9H5T@"
WW+Q&=7]]8^Z`D45.%L,*K9-*IS1C,-SI6,*A1&[0(1["K9!YB,K5E''&2\T
..H>V?TSS<J]JJ'12N:>[C"$""'05*CLMX>'.%`OG?"EO/[DVZ6YLPBE>*BL
`$A#@6Q;JL1ZTCLS%"M[U7TIGW=GM_XBF!$3J*ZGI"!&D:P0$Z1W)*-JHO>\
YTKXX3@#IR%[J,H$)CQ^ZAW)Z_US4LI/&!9(,8I%!6G-<(8*?RE>#T/>IV+/
\Z?3\Z.N!@$O(._3L.W%I\-WW<._:#DDU'Q*C@AF/@$RG]HB`V!>#D!O5J++
9X4I'XH+93A+4.5,(A+:0(!H.UL:R`3Q)NK(]AJN?=1,`K4W&ZKG,7(9C:6,
"6AZQOZ(6IY$J!\[/BX2%*Q6:?#T=!]]W.F=:.VHW(\]KXK0!"E54D!V&J]$
70AUCSK6N:5$W<06@"1VM]\_CQ:EM2>EM2./.X*[RP0U$T"Z$3R$A"]@ZRQA
UT0[#_5,+'"'!KX]SF(L`G_"7B'Z'-&)?:D"T$$[,&ZD,F*(+AI?(:P%PB1@
[%X&MD,%F[@N886RR#2?8$![,M]`(!=U)Z15@NJ7.<*:S./+#1MJ_]CI#?0[
1QU?^U"[*I<U:A>+"$7(UY]XC$T)]-V;>0M$7^8+?"]<325EDPZHH2<LJ+"M
BP<BQ`:*/-]JE!S7Y1@D1#XLM!X..:#@0#D41A89VE5-48NT3!&\+=V4(NU^
GJ_&(WU)E<P;$@DZ1W)&F\D;>O26'+%(K/6%5=CBIO[:OV7U?JJ-P+O.V:%F
)*#44[0Q?XT,U%#S2/$/2>F8K_W-Q&$O6Z>H,AJB.L%SED(QR`S)@7F`RZ.;
E8]+J<W?PYD)`SZ:-%A:*"X,1-S$/"\#X5RL`P[1Q+._GU!AJ?H<Z6AZYSS0
L@PFMZP%Q"03A,.1;/A\XMW+(PDM>'[:#):<#7KH$IR\5_F4AS:4>#W<^6B6
>=E]^YXC\WNS<)6@PL9XROW;2G[7S]]\%8/]F3,18PO@F,6H]S"HTLB>[P%H
].\PL#N//T@&J+Y+O4_IG)Q$MV%$8AO"$%94UH%NN5YI-2>37MYN:;>]E^_0
<.]RH!?E-(?U9KLB54F'&-H*G0;4#<EUL$2/PB7'[%D$P%.M0H($F"-Y2+'J
$VCV0I%X>LE[)P7O_A`QQ0?]MY2FFI;BS=7E3^I^45[S5.0_<K"S8-G'J%03
%^$E"8E>&E)<U1C*01E#V'CBVLY9TI;98.E%?LTR9*DS>Q"*WA0]V^'Y^VZ_
\[9KGY^=_)39>#C'D'#<T%FYPKG`\R=@]2&IAM&/&*DOFC`17`B3R$X860LC
ZH&/-#ED0P*&[Z6+E'7]XOL4ZEN:GI#<7\^OS@:7CPR]"2'A(MJ@,B'">VB4
)/R0Y@394&=Y$T9W.6Z@&4=[4[P8"0DR09K\D<Y1VO=!LO%JPI>IP>S6F_E0
[X2IPA2H%]L/R>GGZ1TI(YYE9";K6#D7*'M.'5K.Y!HO[6^F*4/1>WN&J_>X
=W9$8Y&,%2EM9!/!(I,.3^G`]1O:PN_7O5IC;\\9.9FV\+&"=8OX1CW+(K[Y
9!'_9!'_QUG$XX5F2><\F`JQ!)F&9OYU$=:S[>93X.BM[V%&Q=^ZL;PR\=">
S8SLV9H-Q"/K!8HHZ-=(]TK$R*>T(`*!U^!I'`M.^6D,PQSGS)LCMJTW>2CI
RR@,5L`E"T4#8Y"0Q6RM7*V4*V6KMU30[VPM34%Q2](`%UE/NO(5&+H4E)1/
RB!RRD2.L!0N'R9)ZB\ASS%@Q@@MH\2N7'CHIB`E!43*">4G"A`1!SJ,QL(C
(6-"G"BU49S&&!\=X[0CVBFQGA37Q.&@@RE+:!D$DP/F=&:P01RZ-4/[2@Y@
M(4?%$I;4+%W,5XLYPI3QB3H2;#8?_N0[A5?(2'715A@D[/*H'#E7J&#Z#^%N
M>#,U0^SG=+^E7.R#\JG(?D@)][[3[YU?75J=JP^]DUZG_Y-U.;@Z/GY,R2:5
M:!V%TVXN+_;DM,,)#PW&[5Q,=50B:6"NL"_O/!X4Y&,<LCUCLTDZ@(%113WB
MD"#$)<XPF:`IZPB$/:*"?%8U*C1BI/^P.-$*)KB;R5"2*)<&>+S@!L&PD`9Z
M.9OEPG3>(-B9,`!4&$F>2]<!>&0D[XC3'/`,N/<X:M(:G\&B</?+QFFGC2ZG
M0-H;6-*TT,`;D\C=9#X3DLH$-4X\^AS'E2U2`[&O5R&:5&B:$-+O%NG<)2J$
M;(I'LXFZ2[93]:X)O2J^J?W9WX4=I^\D<.G6C5W-&#M_+*'KT54S'BWXQ0M$
MWU.)+0V?.(K3KH%U,J(IXIC&OO)@H'^S?/]9G[T(,,^7KMA9\_,C*6:(`Y=&
MAZ2G,%1.S.LB!&%.8'U%*S$G&-,P)'HK%=01SKJ,Q*31O_BX,F:5I>,/EBC2
MN0$]I4$+HGF2F"/)RYK0KMLQF.**A-[NWJ-%%<ZS5.+"8INSN;<$"J=3Q2`7
M*IJVK,UPZ4Z$Q<9Z3HZ%12I/?9[GK&`EL>SU`-H1ECU/L8%FG\RJ8FWC"Y75
M0R6MGE'>,LM8N'+Y5`[,,Q^+N/->W@H[3]0]"M$!\\O<'/K<7M)F-@J![`M/
MA$:W^#N>,Q19C6PA]6)`$,<EFM*,J!6XDEQ86]<*/U),MO2.5U!KRM%X:T</
M9ZVBR;RV<KO+Z7P78S;E%`Z[MY2TT_0J2N+OIB^S!`(JKC/9P42P`O']L=#$
MK(<WR1/I+Q-MD`6DX'W+@HX%ET,P\XXRX-\0;U-?YAP\(!MA+4*$?Q1[;SU0
M/6)ZF)#UK!%'H9M/"]3NBTLI!K:1.S<Z5!EP;J$TI3&D-XI`$8,#/$B\$Z<>
M?4B)TI"(T2#>Z8"!\/*SP`SG)1=0$+(%"%`NS4FH">.I$2=,P#9MM#<?43F.
M4,67AKC83L8)B29Z'?M&S%OG<'#5.;'('.NT>S:@(!'K63=]88:H7$18&4)/
M6BQ6<Z!)-L'C,9!?D1?M]G`U9J0\''AHV]:."#$B2YB)$`1T_.7)Q4:A[%%F
MS%7`,[M4Q6.:8AKCSV[O;-#7HXP`=^;/5C*D@0@L@L5+I$0D*JLYW4\"LR$%
M!D+70JMM!9^(O(K<B&K',#.`/Q7$H(C+#[PJG60";%Z[TBQ$H96WF1)$R7*F
M8CLG(X`0,O?-U!84QX2]5]B&!J%9J.`A6+ZS#'R9P8@JLK!^0-U.@5+!'QK(
M\+.%>*LR9!`O6#D(C\6YB6_1&DD#H362NZ&:Q)\/HO(U+@"2.LL\9RA2(`V$
M1C&:;^9X;=%<;!=*58%]7#7PF7]$U4WL[H#M>DEVNN.['0*S-$P,6"RE"TLU
M=SK::0Q`'B->XIJ"M;0*XVB1<M8,^-C*1S7.L?%++(R3WIO+7$$;T`1J0D4;
M385,0!'4C?.A5#5G"\9GSFC,MK,4T*UPHL-,HQ'U'?9UR,B9>+&W]">[B]&J
M/&*!J?=R*D9FEHC%3EF%)S2!I0GEJ?#3UF0V2[JTKA9<+-</TT"Q&YT)Z0`C
T'EH@"U*46V.`>L>Q\7XHH+7#4B;QQ(BCV!B?VMLO1%$*(YC;;(9P+C\W/A(
L_!5D5EB7%(B.(M<&R;V=`)V>A`'[L5Y`$YM0=#0MS`-O7':\L6E+Z:#]":3
3QA?-UQ-/;T(G&?-63K&?YF`RNEPY3!,&E1Y`EO9B+0?CYJ/J@*I$RY:$P*X
=NCN.EA@0%E$7I,M2$&!U<;3>IW"E6(:J`Z/)XG++@GC3&*Z"PCXVD>#PMP$
P1P^R2&%W?3G&`%(.1VCD8F/"A&QFD'$N@1?[8<WJ&5F#%]QA4E,+>)Y+YRA
@04M30[*(#^^%&H-/@X=82<DT.R+HA"ZJ*%9P_)8WHL3,L3/S6.^`N*@NZMY
C7[]7(52.(<.7U^7-)[Q747*J&-JG!%!66YD657T,2JP81X$SU0:;9H9]$,7
%<J(`1*:]AE(:&`EHV^"9T)!).2%J@J-Q-V()JZ0^87L-6)?Q7!'KXW5KP7'
0N=26DIZP[@,,<;&**1OM!=J;.(KJVFLK,-@@N8>QM(1,D$H\)QY(%Q?1-.@
EH0*Y5EOI"%T,'YPU&E5Y5N/E42LKB`#'=Y=;#@6[SGU$0I%4J!UZH7<D15]
R;7T)90Q,C*C.2SM@GXTZBCMGJ[6D"[#>`6J)%X!$LW;*JZ9^B()3N+%JP5M
@B;%2*8HVK;Y#(])2P*$/J*>Y_,E>3CRQ:4$IC=XAF(4F9G=IS`X%\'7L86)
BC-W@OA`$K33I/SI)2/!D%^^CS,P\,T\V'36/>K`"0A8(7O[DPY>`*%.T4<>
UL[8OR<@3#:M"@F!M(-&7C['9W9$VT?(!2QF9>OBQKLK$PZIKN1$-"?4Z8<R
``\%@_3<LG5"9PZ0U>N))T]+BO:.UD%(A&%E(^/$;D/_6#FA7P+2CWKK:(CD
("!7H(;J!UR'?XM>$*+C@9[\A<9[EQ1_CGH(=1^B_/E#7G"F?F+XH!!C,:K(
:.%,AX3WC@*CZ(WHNF#6T&(DU'3`PFQ8#;`5<B@W@B`.D9YFK837AAB1<O`K
KEAF^9M:OQ]W=F@@Y&\8-VW8J@?Q[:K=6DFN#W79>+'"JP9$>M^5T57\B`?B
^Q2^I=&H(?IBTE61/R*U-%\I(`$3BE8..XZNV3X0KA]9;^Z'$EJV**\FEG'>
++"F/J)EL1V9C-U0-,0,=6$`G#E%;N!+-8<4GWA[D.!?LS0Z%%C0I#7#8DQG
4DPH1XIQ%8K&!AOJFF6<28T.T=A\(Z8\GX"OE4I'`LSCY`[%*X;_PU21/D>A
SF-"I;ZA,IFSPS+C/#.?`WQV`OU?LAXB&.?-+A3H/,CXE"!(G_4%)\BE4!OX
KKIPI(#V@F,K6Y<415X%UL&+25:"*2AI%=R=_.D>+,1JDB;D-)8DJ.)%-EGE
$/?+I[0\CT%@X[@IL6614`:*$+.LI5DWJUF!4[].U,G@SM%UGB5-ZL34<Q"\
&<DM#A??;I`>1UP\ZU>UXOB-,PU*>_0M,V[Q>"K^Q90BY+(33(<,Z&GJ`YD1
*M/V*ZM;.K&^B]&2UMFZ_+/EL!`%X37BATHV=Y>#S!HO$]%$-ZQ7JR:*[&E4
%'O]V=`C,'<*RTALC83QD8C1)XR#Q*]-S8WJ[9%7;]3WZK5,<R-9HF9GM-]X
"KSY9&;T!YD9K3<T(@A8NDCUE[)LC#="T0N0>:*LOXM)T>\9BE-`3\<C<:K7
AG7)TH5*3>,2?#GQA_321*(G7/F5#BN_DKCR*P-67C$?*X*5YV&F#E]=WU#8
A@7;)\W$#<3,);Z(C$Q107U-CE>J2?!SB1A,#GF#B$,7D@_)*HA-%EH-GC`H
\_EDLA*:KBDRS112%&CGS$/%`JT+"A2'JDJ^YI(60[T9VP6-A)N`:A(VEL46
R8WCPGRIM(G$F>'_7M(';`&;VI[/I".?D+J*9G/S(*8XPL=%<G$%NI2_GSLT
+"*416I=T,GO,1CLKJ"D/Q`;JL6((%R@$&V"91=1;H*-W&J8C?))\<M0D"_C
T#X4PR&,->$E=_`LN*/<9'XGF&<BD/YBM)JB7=5(11PC&1Q[+=-)8VC$R,+"
&%V=_!<P4HN[O"GB'=1RVYP-A=WS:8961L!UHQ:1U!Q(EA$6B[?U$'U28LT&
V9#\+&8(K*F\WN!?,6F04VVK>-!+M<`3<X$S>J"B7NH;@?AR^54%NTS$NI19
Y#ZS0GVCR7UFA<9&D_O,"FFC::^I5BND:B,*<=H[,T)R]L[RME.T@6G)PQ\%
$/[S^.//_,\K"U\6C!R=#YDY',XQI!R1E3-&$HQY#?[8N:BV\O8]%*)L=H""
6#9#J^(797^3AP\%J`L_D7QM_6;Q+Y`C]PK*``*<0KZ%>B]=0KV744*]I-=0:
1A7\4QD)40:]-2^LRGVE<GR,3$]62FJI2,EIX^WFI8971R<G[T_MB\[EI3[L
^*&?Q]ZP\RF(E<\M^%U(B;.:F4ZM/:.:1.R5?N_L;>_X)XY.<M8YP>*^NT])
@1]2D^OE8?#5-\`7]+I]:`QUV;:=<`KK_I8P$"9>/I>S7KUZ9>4XO%&.S9,B
M"WK69-%D8F8R.O5G-M.*_+-G^*5(5DU:)F!\M6R9N=A<*VF3+T[+-)-\R0]O
MS@FWJI7FN#YT*UX6)VSRP%4./J^.=9RO=[9VH,L7"I0[S3+G0/N>9F%TD/A*
MJKWD:V%/%'U(V&CIAE:&]502-3S%NB=98\+`Q[20T>[R#]9?`6X7Q$UM-,/F
MH"8FF%!I$#&'_Q"SN5]K.Y6*X^S#;#IMQVMZ;F78,C%R1`:>4_$#)[7=0!P<
M^#_!RA#@LFJ,</TXZ9U=?;`(IR;5EE<IHWEXT$L#W:IL%+KQ6DN2&RAQ22XQ
M_OB!^$EVOD(WS)$PCY%7?[CK1;&1:@'=LFDFYRN+\3"D_9OP[!()+,D^;ZFC
M9NK<VRY>#1R8[QSW%CF``P$-4VWLT7!4&_O58EN#AF%S$X+[0*B=Q8*GGV^Y
MQS.!WPX+3^#H4$ITV92(UB_XS9S?2H0+>GE-FC6FB#8Z",ER43]8RY.J0_M*
MJ-A\?;-:H,Y/C`?\7Y2*=G2.+R-+DOT)F>"2<3`[&"HD.3E>E%&H4,@]U:9Z
M<77#H6W@CD1]PUKTU"`2+.E+T>+^7D^"H3.QIR"?P8G&0]RLU`C)J%FM?-,0
MHYF!,-M#9S49H90N^&6'LK"!&)_>T!AI+YZE0PXEQH"Z)708MA]P4PWT$H:*
M$@D3V"9:(8DB>*C:;5J-S;UFL?8M8X6/Z\%,H#(`6CR?K*ZO8:$A'5!-0GRD
M2%>4.0:*?\#MC-IGV6S:@:N9<GK0MK+,LLX\4J:!NEFWFJ0E>".<PZNW7'31
MIO,S%/-V$2SR.;YRCI=@P/>D%6^A'9C2F/$,ZC1'K74C#?9%6H_0V94O&-]A
MH"9H<:K&Z0:DNNL;MAN1!@;&HA"32NI5/[!!8IHM\WKCPXGGS?,&7%"D\Z15
M/R(':KH%=X`%.U"5/$Z:HF3K5SA:S:QF:.P&4MK"CG]EU2J1:03?HA6-P75P
M\[?WV\5:+7M%9R_G!)C/=*CA^>A[G6V\S-UFCBA9V#,M4!;RSZ@<WB)TO8CC
MXLQL6-F47@VYS/!X#C0LS*`I!T\[;OV.HWD-"Z4?EO^XL6G[&8F3GR5GH*?*
MG/KL_61I!S0N2!=:M0@>[%O(*#++O9_<:P*QKU$EG+GJ/I#O*BUV6L@X$0N0
M)6X].WR8Q>#Y"H08>$RP#8@DC.Y@'-)+Q?^8LGH(`R+20:=0:-`[9?"NW^T<
MV6>=TRY?ZU9;2-;%+OLK=&?`2V>;<0ABNPV'S=B2:@"0D>#>Q]LKCD6V6.,]
MBVW[.;4]'\6X:0TZ!*IC-$>Q$6)O;MYXO6AD1XF.DC$NM=]S%9Z>`#C$!"QU
M`.$2QVZMPCQQK5K=+]8K<2)%(^*NIM,'(5BD#DK&0UY$8DS^[WGOK//FI%L0
MV(01)TWUH%R#F\0FAEF$=A&D<+2\CVR=:>!AC;"IH'X*B.F`SN*%-=O"K*$7
M0`>(@B?W*1ITD;FT,F_1J-)\`4,]SN>($)`IF,^"%\&[TM5;'#HO,JGEW29:
M+4MBKG0UQT*TANJEZ!Y4T1M-QM+(B&$H)(T.[PP/D&=J$%)75D14,HF.5M^!
M0=)UH;9`0(LQ5QGV#!Z+61+Z5$E<C.P:7Y&R+V,])@C!:,4^NG]/6200GX1=
M]N&[WEDW_P_7N[51IA9B@[95]NJ\59JU8K6A2-PC-2K"*<0+O.!<H&U]]$43
M>.#KXW)2-<]RD1IZLBI,K&1AZF^N7W&"R82T8M<O5S4#"/DX=FWA=9*Q<U)W
MLK1#)UA'5@-8W'3)@SBA(7*@5<Y8#]V!&USXGX;L>X!RD+"VL<*Y@YIM?QEZ
MDW%1(TC;;%8NHC-)])%@YI5-58,"YS7@>.M>>\_UZL.]<;GLMMQ&;=P8NA53
MU6``\&J0N[5ZM8W<G_@W1E<7SM0F,Q@0RMU\OW,JK`)FWAV_+UI=HC#;VS!A
M<R!(03"Q0B!^()_&V#PLBZ@#$NIY7A4!(FJ`C*'V`K4`'.\N*9HY[JT?>AMF
M+S)Q/^T<O;??7;WM7G3>=F-(EO3MZ/QL@!P57]J$`5G\\*W:7]Z?(E5RAG3'
M0_@?P""`M###""BGIU=(I4J_4QME.](@37^74E7/HYEEW0]-\)?-R6<\(HWE
M>>T.P^5J2"M4_2T6Z;#1KCC>:,]SRN5ZK>%5ZJ.1%\.,CO+P.HU^$SITF^@:
M_E.M&1#!L-DPO+O-AD2OK4N$3["[=)+_QC_.SGO]OZJ_$4^M3WR'*./MT1M!
M#;%..JUL!?9+7H31>V$8S]I$=>(+T2J6'4]F<4JQ/,@[!%(AF*E]/YU0*X0R
M7^CVKBZ[?0[C8XRNT/+1[E5#@YKE]`]2J[Q7;]7W6_NC:KGL-*KNN%&K-=OF
MN&<4P).0\9'8Z1:I0EH"KUO>)1'IR.-((@XOAH.T.0+7TK-'PT(J9/,FCZ%$
M7J`[O_$&*%#AL>DB7GIW>WN+:"XTT`UD&U_1N_\S&KY2X2A$[+5)Y"//EY_\
M$JWT9+]<<ZHF:#1;PBA[N\I=&88R[;7<'FZC/FKL5=MNN0S[Q*DTQM6]V/9(
MS<Z3E/J)Z'N->(&:F*)(,NF>'-OGEYTW/1A8^)O^1"3D]SB"0KGN#/TLM^I'
M=/@D,9&5-$:H%($]8+&2PAJ^B;-GCP#?:_7]2K&:8.JQ%S;TQO:GSK67UR\L
MZ`U)-&S7QK_'KEA,&$&B],-P\8F%:NT=%B>IA'?C+N"-+8P.-49%J!NU#A<2
M\0->RQ+QG<8KH$"8]]F3VK>^5[7,;V:K*;S:V3$X"&&,ANV:0TIKVYO/J73Z
MM6/Y&LW&AM'GT@]SFR[%@7&Z&-@GYYVC@L"DKS7JE>(^C&FCWD))]W<;4Z,1
MV-I%L+1>6!?]<X0ZZ1X:O=(3WR+W`P/!PQ6MF=0,T4P97F]4QH&IY-#K>&8N
MQT+2;TZ4D,C]>4VC;:!E/ZAI=C=IM3*"%S5"$8\U.RK9L*+7"WBLU9\?GZ(?
M^[U!%QN?1:<$%8S3*?5:T"DX3=JC8;6^!W2J7:TUG7VW5<FF4U'V!)V*/M%1
M4B5"A?\PI8HNM?[AS/U=D@?P;DO[,'9+9+]-KY/WAADN_9:\)#T\.3_KVKWS
MB";*-_'AKMSOB9M9_HW7<!SX)Y`A8ECK0[=TLC^M)G0%>]1J%VOZ3D2R2E0U
M.GRJ4HE`TN4M[T`@&,4H,1QU56TO2GX#CC]RD,ZC;8J]1#OB/'ID%RB]/"&7
M@8T<G50OP\M:H5!0FA=R^1!!%<_Z]O(:BWD5L;49E5W':Y-_UHI1WL23V:1Z
MH:`SOO_4-PZ.!_J(H?I1U1??6<MK<C'36A%+@)':M01U0R-)\WJ6\'#U6<`;
M>NA#3#X'"T19X=!."(RY`/D;<;(P0K/I8L:;,1]W>'SQ(ND#^=H27[")\.NR
M][;SIC\HQ'O`3G2BHS&_26VV^&/*9)%+%6;/6AKP#T]#5.)G0QCYBK602@!E
M>8\LAX@[CM8G2OB+8+B"Y.@#D[J&8VE>F82/HA8&<X[`<0/L5_R-O,5OU:K>
M?KTQJH!H/1ZWFR.OU?1,<A?/R90N_I;5ST3C]N&`AI\@BWSH;6$0:[R0M&Z\
MR5R`#?G0I2Y_W=HYZA[G<])"(U?$6*LV]%((>.<7:"]KR^]%:4R4*ZEP*V/2
M2I."D2&I.&B;<`ZEK[_,<J*\3O_P'4:%12,7BZJ^9=-G2%`QZQ0?!$W*E<1O
M.;6NZ)=\S6&]HK!MV,VT>L6X&-/%:V3764QW3YU/'G:K'`S_'L(X9WT2$]@<
M.>-1U1NV1N5RK>;MM]L@"=7-"<PL@F<R\S,+I2R3LNX$7I<>K)W74?"3<F`%
M<QOGUEN4@ZTHQ6CQ,,<UG_SBWTVG]TOU!09P%/V8><$L+=,T>KFC7E+<)$S&
M"?\C+P3,R_/CP>GI58'2A".?DF0-N"AW9(Z$?"O%EW9C5*L[^T,8YJ;3'KG5
M:JO9RAQFE3LQPNH+#2[Q!/4D2T"R*)Z5$S=<,E?`1Z]A#</=)WL8=<1W^J<4
M,>RXW_VKB$U.YSI>\/0^G'9?657K[;M?BQHNGD)00/4P"S"*A8@+[-CJO7J[
MBNW>0TU%O:T4L-`]4E_#J:^<Q'59."S$3W?="$!0RW>DU>03"?U5=(?&2'P9
MA7CMBHX'?,V"4X0!5#X<7MAO_G(Q,-RH+>&;9(/`!#F+Z/H2NRS?M811CC4?
M"9=>;)^FC25OF1$NJ(9VA)/B:A)<D_L)]A5DE;=H]U>T<C\_^TB1^^8CD)7N
MGT_NB1B(@G0^0/EBB;,H;2QTMT[$2,#X1,XP6"PW&Y.+?O>X.T`*].:\/[!^
M^TT_LM9E/.H,.B)3;$R%FP`_VWB%P0$IT!=,H!YP<`6@TT6.*FB!X![018U0
M,[#G/\*D&V7!@(8BD`TDPRMGPCFD4$O>PM/3&OED.&6T<,9HZ9['8T0%ZBFU
MNVI2LY>JYGH`GD8L&<JO%DV*(I^'0BCFJ5=V>+-:NL'=3-H$Y"_?70V.SG\\
MLP\[B`_Y[AQ!-1"+H7!@M"7.AZ"&G<LTKN/UNBC1VHK^>GJ!:H]!]]1&/)9!
MK$ZM_W+YR7HNO:F/+`L'19\ZP#6.0LOUT.H)[>4Y#BD;8K!ENG#FV8HFQ)5!
MH]E/@7`AY5+CX(G:Y8+Z0M%#)T553AA84Q&C7[H9XD(H9])TIHPQDBY>)DU4
MV\UFBHGJL-[VVO5AK;Z?:J*:4G!*;&A-RH-9VPU"&+Y8?%)3[YC\A/\KH2!Y
MD\-E`6LYD82MA&.9;^YVPV@&U=^Q5-$9$KU#NZ)PE_XOK"W3+#"W=M`_LG_*
MY!V]A(CZ1SZKYM4RMY'N1I`7Q@G($U;O-F+16OHYP=(AQ6\7;IW$6*WCL^7&
MA<1WZ`99M&[N2"^#_Q/ZGM">WSR$/HBCD58;/P!YPH@&-N?SBVRXZ<.B"C$Z
MBZ3,TMT';[VKK8_:6[+LO%44!+*`>$+-H#T`W=!C.=![("153?91]4/&/9&8
MC=C34I"_`[Z,&JK"3N+K\`Y]@_(\:HI:D?SP,GCY*BK4R'_4>V_WSZ_.CNRK
MB[QL_5[1JNMGE?0AQ;_%-=XKO?C[S.*C(@O6KG&(ZF5R*:N71J'N%[:9(`Z@
MX9DM5X>L,*?"&?N!PJ7J9P!(>,:2T1?:6)@-T-H50MGQZ<">3^ZMW"LXYW'=
M&=>YQFRN*T*68`:\BI<W@=Y':U,V>`+=4"LEUF3*(;\=9'<T?KIU^'+W$N]V
MK6T'AQTY!B>,*?Y.O>G@OD^QI3EJ&^>SZ5*80P4Y(0_,!CLY]IQV3P<?.H-!
M_]*^.KN\Z![VCGO=(V$@,BG$FL*QX9Z]YFSV^5\2G4I,0,XZ=&846V-$YKU,
MKF(F)/+1UY)\M,/[,\UVO$KBT8!`"D*XN+-=;[BZ)N:"J11W)@H$\X>V65L0
M,5,]L4E0V9]<'8+6T-Y+?#0IA'QH4U=?)9N(3O<3=S6TYWD8"KP<V+0C5&0M
MN\B[KRRRD5%D7D6ZG+B3KRQ[+[.Y__CR$C^;/V/+)+$@TD\(HWGF29%>\//O
MMG/61;\7("JN?FK>?M%(F*=&>E4@056XLOMOK&RU0;^XIM4WUN1N7)/[U37%
M)MY'"?4N1N&U-&;M,5)!#!/DG^CG#!R-I>C59Q6B1N?MB*GS[I<+!RT?;91.
M\R9#I[.*VW0!H'@_DQGS#Z*_[V(<$7.%?!WQVJI5DJ]M`K/'$TK>6JC!X+3Q
MV\MZ[<#:V?&3)S]R:/6DJ&>.W^7%*Q3M*]66+MW?PQB$/_L?]:$51T)4=)*\
MFT6?]+^PZ+6E?7A>J;FQ`OUU91J2J%G6A6`47L563QJ#3SEP_FO$"0NP`=CO
M1<FF&TVP2N9$%O7@<RE->>\OEJMD2R#=OZ0Q%;V2S)V4V"*&$0=?VJ;YY5GI
M^\0T`='M.M6^D2:L>'?R'*8Y7@=-M[Q+`?E_/)ZLPIM\N'1Q^YF6HWJP(\,+
MP-"T<,AT`3RH`K@[9MS]N;>XP8!F#L=C1YIH%`)<B^]B5"_$&UDN_.MK`K2*
MM]YRKAU_5K:L[CW;#QJ%J*))F82:'[Y)7:?NJ6A3E669Q#HX:;JKQ0JE:'(.
M_8&]IX'EN"L45&096?4^:BX\"O'#=Y%HH$8YI5Y',ZXN"[Q+3XR(]DD50'&]
MV.V1%(B!S$J!`)4W!P7+1/ML&8N(VV848X2MYPS"Z=&L6X3<HQ0R=K4J2"&A
MEM6KL_(;#.HIX[,ZP@Z:C4QA+3R3*;4!C0$"Q+QW9:HT[Y/J@?1YY73BGJ]R
MH$==HY`TY,Y'6`P4WKA<#G_]6$:(.&RF9+.MA3.[II`=:#^(0?4T_U%CT^)?
M6.2C1Z(9Y&RYB+T)?XVV?-Q4+/Q5])Z/U?%<G7BDX<`#[QYO`U)T&9/(@E\X
M*&BP#:\CHC>>4W33N3?+1[=QN<4PIU&09V/3>4C:.<L,!E<C)J!4C7:C>!W^
M:N@S\-0<DP`I):1JT<*:,NVJT^I+J<XDGU#X*^OYY!Z(Y7:>!A;';KL`[PL'
M&30S6Z2C^5/M34IT\OR`$J'-&PISF7T8B\!5<YE:I)1+X_/Z\RAC>1H1T>*K
M$3,FCB?-8#1AHB1]Z:.MK(7W]F;N02R?%K(VOE?3>N(&,X\[8KR^U=5]:0M<
ML1AT-F#R`R3\1%MUY&.B?KC'!;7`Q:%=+*W?W%$3Y-8+'V;N`CW018!LI?PN
MJR\8'>/X^.#+]S&+=U9>EF0L/#HL*S')A**U)TQ*"ZDZB,<NO4A,$2C%4&:&
M/N)V2H0YW[\ZLR\'G4'7/NJ^N7H;3QJ3;^(BD-`E6+%<<EQ&4S0C48$_U(":
MJ6&KH3>G"+_/F_9EYV6:8HEB#$*11`9S=">PXN*!:IS;_2-D'I+JJ/S8%<0J
M93RE+XC8_U@TWS9PP1GC%R<%&0-FQ`\4VK*,:RR]+5%P[=^M)69HO7@;L\BH
MQI(KA[!T@BJ?KR*L&_0E)E\9]QUEV4+:H'MU-J?#F#FX`K__WJK6XK6E%D"^
MJ4:7$X,E=U\^=U]Y)2Z6E]9]]55TR1SO5D9;-TM7_1A/)@.(Z.^,-B.?D%[E
M06:N:F:N:CQ7*@V(*_^^]9)T@RKKL2K%VDE(>&(9Q]9R(5Y<7->HV:'1J99>
M3-$<Q$2IS76-_(9R6^GE1H=P9F?3E<()'E">]W%I&DDPDDB;[`9H)S#/F]0E
MT=6;M]#4V60)[OH+\A7&%9>C4KQP-Z>GF"^\L7_/"<@"P/B*C"454*WMR;4I
MS-RQC9;F+HZ4+8\O\Z)2BMJN"#!&"[^T>Y='O3ZD*H=+&Z^DS5-W=]<2;<2&
M1W'>@#\)EU$RK&GZ"5)$->&]=O($Q_+(\@<+82=RO8(L8H,'^R]JM'[)85IO
MM$1!B&CKK>-/4!0KEZV!L-8`%B"%RLH3!@ND!B=2?*TM1>J"-"CYYQC7C!%Y
M$;`IN$.;4_B1EYZ1-*<S<9+("5?!=^6+`FIPP^=AZ?G$+3UWRS!2P`:H">!E
ME'+9-<,`@6)URMK$+8LXI;0ZC^G^ZAFA18DK/9V;XU)$:'%^OK7AR0;3Q'%O
M4AO_.5KP>2N/.U')BE$;<G<[.<:]PE%.Z8NYV@[5*B6.B)8>6C_0_GL%B_%Y
M^$N.5%NJ5]H8Z,N,&F(LLV]?8IF26%Q4T<0$-B;:5%!`JK;-JG45NXR'V,K'
M".`+E4J1%KU],J?N#3.>+@7(5VY;/81[]\LO/1U#`-O\3(-0^T6@<&8\N>-.
MWV*#-V),NB?:S\>R8KSP^XJ>G5]5]2(B7;Y8Y:KSL!*X5\6X2'4K+J,WN'@6
M!GH;I-R4K=HP9W3(JGWPU=UYK!-?VO0U#<Z<"]B;VLJ"=6Z)5:XO)=EELAO%
MO%0!\21:42AC'ET!CW9X?M2-]%&/5'F$!DRA-QU.'M(K%7H",G32BU%RQWQ4
MJMS7*G"4WC<J&]=[ROK!(_B>7F_:=9GBPJ*MO/G@=M$@;XR1O]#:,5&GH2)*
M*=\@6>M-ZF[2+-]2HCZFF]35O"%Z\8_'H\U,ZHP0D+$(D!<=M&<U8D!&K[Y4
M.W/PY72:1B\90U&V(1%%<3FZQO_8*%'^+:W+O=JH7A\U<(S<:LT=UVK#1BS`
M091'#);Z31$56Q4*J0C_:%%_9"@:/506>W5;?.OP:+)_2O:6>&_R[9;!1SAZ
MACJ2GM%[@]VD>";+>TS+?\5$P+CA,ZU24;H(:D3IOC1*"&4:'+X]E)?`F`/5
M59-),,H+5F@[5$D-CSJ_B`B;QM3=LM?@[41S]J^UAO5:M=XHE]MNK5KSFM7Q
MT)RPV\@K\%9Z`=;9P[^>,/@'7@3M0T-OA![7,3]`\1%GV_`$S`IQ&G.WOSPZ
MH5<B&JZ;M^W.Q<5)U[89_$R\Q4NM@E;M]Y`/(YY3NRN\ON#?FO!DQK'"+(2.
M"AL7@=NI)1@S[5;]@*TRSPA-I"*<)+ZD.*V:X!7R(5$TS4\HQ?!"@V?.<^,*
MP1Q)3HH:2-1FO$O4)1R`7B4;.W<6H2>_Y[F6M'Z*:O[T]/S.S^+3O(3^:<BW
ME*Z=2;#P^$@!V94<UQB>9C(IL1!`T!NU<K,UU-^6*"F2@C)E-NO`(ZW5:N"_
MU7:SHO^+?S:KS>:?JO56M5%O-"KM^I\JM7:CU?Z35?DC!F"%"A[+^I.S\,-U
MZ1[[_M_T2<"VX$(@(A[]$*3<K=2;K<9>&2;?<4?.V"3B6O((HT6\X&!4U7T*
M+\3_PJONAPO[<L`'*T6CTC6!0':`74,"27[VBE+W3WJGO8'=N91A8BT,K[0@
MR(Z\^EBT7BSX<JIW/0O0SP9EW%`S<$#0[2_*9RE.0'2GQM&2^-\O[$Y$:N$4
M6[CB"FCA37QR+6.W3."E1,/0MPC.,/2?M)Q9,'N8!JM0*X70\$KH_''G:>YJ
M<-J32\IU$!`R)KHFDZ2*9B)P'IZ?GUK#U36[*F</)];^=0/Z2$XCFO7?U"3P
M)Q[DNE@S]:]:,_@LRM@F8&F0NQ(_1BNAZA2=3C3\\+S??:S+:1U^))^E/(9Z
M(<:O9X^U*$ZRJSP,*?;SS&4X>H5A2E"^P6K)I[,?+0$!<..Y18M06=#80LD,
M!PQ2=N<C?!'E=LE['>,6*,\Q6@,<BJ6R1X$QY1]1M(+5'MJ_V"Q\Z,-=)'@$
M`7+VZ'[]RAW[;7LVN^[-EO>_:('7:Q6.N58SPVU_T5!;&R_TKUSJW[+8_Q7+
M_?==\`WFV.G?ZE=.`DT`*P>..U<G,#C]SN4[$56#!#9AL;9M61?!Q!\]6+=^
M(.`F\Q3!!T$GR>@"7A6*(#1ZSA*!<T#4ON:L-!O^.,]!X%!_\(-5Q4@1/W8_
M]`9H&G!UF6>H6])NUFM*R#3:-C@]OQH0+Q]APHOJ)A,+SI?2K]Y"H>\1U&G(
M5]P4,E2XI)(](<)3"H7SES7OV6OM4CIMY'94G%\$`8_`Z6=.N$2LH-$G,E$^
MO>R<201;B>H9KN:DWB'MA!W,;(Y]BNA>XKQ$%TD$9W-X(3G"0'`D(<UQ7;3V
M]ANX-UO[E6;R\"%L)'OHSYS%0WZUM\W1$]2&_,P[CHQ%O"6(=OD<8D9<_J5W
M8;_IG=F'[[J'?T$=OR;98V)S[$#FW"RW'*=#;!6-RQ#6EC.C7B[$IBQ'=*?=
M@@.UNH]JD&:E6&O%>\?1&3D`APR?&;*F@WL(=?4]$50:-@15B1*?9"7$#A)[
M!M4.U#-,2Z.%MPB30$C":E=)`9D6QKHB,*$:*MB-&Y5(@R2^8CA#:OAK"T-:
ZNH.2E&`C8J11T;6CM40Z@\@!]L$V[X]PO\MP@G\'ZM%4TA!8A$;7@&#<6`8
1J!%W%P?0Y\)/#(:J))!&#;,*%(+8%$!18J7/S<!4F"&"V487-ZOY#O.\4=$
WEP)KXN@H_XLAXRG"(^/8);TTH\H#A\9M`1)HC\Y?YLK6CF9F^W.J>M3;SJ:
/^35R.Y8]:(E_JJJ>S4UNC0K!PF2069FDBY\79'4&IGGYQK>30O]-<BF-P?&
5S*FR)5*\MIZ33;SNYG1]"Q(-KNZV4A$!.\LN,-IG`D$1F>$UON3!VL,HAB=
HK0VF/P0#!D!6E,E/U<^&CN]22QTN]5NFY'UOF%O4BN17,%VX-_$78S0[I7V
DBTM19^')`>2>F"Y<$88M@8S18:PL:F7#,>7EZ7[C:;G?AYRCJ(<IRCL%79&
W"?CYB9/2*6,.P8N[R2?NQ)AH0.>A9?/PY>J!3S4[18/]7[;9)R_=:@MHC8&
)[?A^$CR%%UA?/.0ZT5^P;C+;,;@4YY/]GCA>7FM7&U2GB5F180W;+?V6$YI
MZMU/,M^O^$6NSFU826S67@:7W0&[ZQ<<J!4<W>2ISL./`-/?$$Y8E@TJ@2C
K2@269-!F\/EPEW-UY0H!UYP7;(``5Q$S.J:>93EZO.8RKQHO3,F3UJ4K.O%
W+1FCIJX(YDK^.>R\]-Q'NT:K-%)_\C*_5SZ:,'?_<L!+Y#]&MX>[+3WFQR6
-DL1G]092+?$O[Y\A<28:"WU2]>3Q/I<D&3B=#59^G,@%*6_BE.7CV?!E7HN
3D`L3#2;?>L&.+N[WU#Z3K+T'0&M9.F-?P8GE4TRG8U1QV<%2[VPR"7<9DD/
!T""<.U58+L!E=NK-%K,%F\PK)8PHR-@;PY(!Z(I_<S+_6[VET<(N%?)IIF[
6:P5=:SB4""P%;%L)?%3CD1F*1H+N6F!DN!9L8*-;`?Q(%HLY]*=L/Q3*%5K
K5I[A!&9QXU1H^G$;L948@$,)W^2W$I8/?1_CN<+*U6"4S-J[2NQ7A4$2;=_
V3L_L_0G1[K\7'0WG)VH!%L5:((3YKBVKWHHM/)7/!PGN(4!P';DS=J6I6&+
RL6J<C0KAO;EF0F:"L,";X]$?'WAJ,0KGT(\:_YDT?;/G[XIO!*,.['M7U5"
F3(.!.<N>&_F^8/5PL)K%2EC(UB!(QAW'4S&W)K<WUJE(E;!V6HZ!*DO&&-T
:G^X8"W#Z&$T`3E^CO(@8OQ2R*W(*3#O"+D#7XFK3#0EV*E7ZF*),?2-=%LD
_:M4@H!(373)]Q:OA,L=@=P17#&D+X7>A,!Y:>/(3/,)HI[@0$Q7(+]+N&[-
-Y*B4\UA"%?S<FPU8QQ[^\<.C,'IU8F8]FHE6LCIW]4@'6I#(]OCN']?A4N,
C(5(RXA\"%(8H@E/4;5U[<V\!8;ZY^GUPA5B_5"'\)Z#@S\))''LR>)!Z/7E
+"@_T)+CNGR)[BU`PB9^,BS'8[SOWMRB5<W^7@E-XX?Q-YL!B*Y_RF7'\1K>
R&UZSJA9;>[MU2OC:L/=;XRK[2:0HW%SW!B/&]6]K;>PM82\05=\6R"->@MH
?&/K9#3]S[]NO[Z_&;4J_V]Q6-E2WRI;[T;3]W]VZ9YOZ^GZ]O]_SQ??_^JW
OOCM-BQ%5\*4HU&NEBLEY$;$9?#Z^]\V$*ZZ>?];AU^UI_O?/^(Q[W^!79^0
O11"W"B(F;37@M",&NWA7JO>:+;*Y=%^K=4:M]U1+`AX:G9Q1YSVB>/CTNUJ
-2Y)^S,,S$164H;O(CDP#VTL(08J80U$`%:@RP*SQE\.-4,;84*C5#CJ`ZI<
2>>%1I679[V+B^Z@)D0OTY`['1M0,SW/`#K%)W(_%4!CV/U&A4+V[S2J%1-:
5/0_V2?H/)XBJ8@:::D/#%E2#&)H#QV\,YF/I+V7Q(@7R`@[QLNYPD7ALH:H
,AN"A!U\6LWMR`HV3_/P8@Z"P0M5QPLJ$D2%,7E'Z@H**S\?D4/U_9M*M7+T
IH.(9A3H\=/<GM/MB:UN3[*,XB_.3WJ'/UGO>^<G'31">O;LF77)7%4?O9SB
H&<27.=W\80B@ZVZ\F3[G-6W3N/P4/3M=CH%87CFC[(Z]/[TU+KHG/4.L2<_
JGM^%/Z&'.X3/OS+.Y42`),4@4/IK6#8-TRGSCR)K"L7RC6:Y@/OPNL#EX=:
'<G%(3,GO*W12LU\6>#]-KNUYTL=*X+7J_0;5*MWW7ZO'L0ZPVBK1G=0Y^]A
2"!B7@=O9!2(6XZY@H%R;_"";$GJ0\7D.B&&6\6MD@847])$1OVGC$;>]H;[
S:KK`=UU:@W/&^[MCRI)BYR8`&F\(CI;;*+%<%/P]RAU'P;S!X[240-1WWH;
!!AIHC<;E:T.!K3`3R'%7`4&W:7KB/0L)R>'Z3F(7ISX(V^&D4Y7T"$>DL[<
`;Y#?BDJ:;96KL`*@R\Y\4FHL!Z"E35UV*T*;_-(DB".F!T`<"((EL>G@,5T
#[*,*BC_[Q6C^6*SH??Y:T5A;6"RI-*==0GRU4JML5TO_'ZR*Y:*KT,.J`MT
\LZZ!?:-MM_0NW'@[%APXPTQM_H_2LS];R/H*NTT4FW@;Q3),U]*U)EQ93AL
N4X;9-?]NE=K[.TY(R=)^&*9(_(7^[`)$6Q^.1%L_O<B@O4FV:PW%4&(;-N!
==@-;Z9HWVX9P8W509)+B7K\"&4C\O6^T^^=7UU:G:L/O9->I_^3=3FX.CX6
-(JTRSN-MF@2/&C_>(/1%L3Y.D./!=Q$GEL*8$'1'4U9VEW)*/]IA[KE!L!T
_<(Z9F3>!4MD+\-)?BT?(D$UOB!34?`=E-G2H,DKDN:1*C"<^?.YM[0^4;PF
MZ*)"FM,CVA/B&!(E((9!Z.,[#"?&"KMF`_&^FDVI';50NKA>39R%*`,W/!,"
M?_9WHE/`I#C8F1(&L-YD[&K&V('L`\(4Q?L']B^!E/9/.5QKT^%-4843BU-*
MCP=FOM7-/1%(5U80#X8C<WV6?^!GV"9#CQAJT9BT*6'3GV*C9NVT*\5F-3HA
MI<86IMV_=288/Q^OYL7@O3(M7Y2G3.1TX@!)%!SIFJ_REA=K[-[SG8XZOG#6
MYZ@29D12"BJ&IS%K;%P.Q:5@_V23(FB=#`B_GW]`X#^+WO$2L_(\F`7K^X];
MI37H?]C*K*QH$;:^XD=R:XQ+'&40LY;)9XDG76X,YG(2614@(;Y063$\OY&1
ML[*+F[Z@*@<Q/FI71O:3$?WDP8A]EKEC$=BT0B!['&<:N0(/2#^S-7HQ"%",
MWMS)9D2M\(!6N%,OO$:_[VO9$%AH4E97$0NEV`,?T[RBT*IE=SF=[V)H"O(@
M9H"#I1G;3AIQPDLM\A-RAFOW@1DW,;$1S,\B&H4<AK1(=;A3UJ.\\4;JQ<@?
ML4-9+97E(,VT%\*)#L3BWG]V]1:O3<;5'DOT4-<;`15V&`+*K)8!813!$]BD
MB:]I<3"U=$(OI))'I$[M712B'TUM1%(_2&LF\KMTY*EFD`5%6IOU`U(OV(R%
M5E3(`-%?V-8-:MSYJJIP/\0KI0C5<MK0MU_$*4'3Y#F:Y2)H#YGF";*K8)TL
M:1XIEI,,<(*M@M:P-L2,5\C:M335&QD&**"$2.F6HH=+:N&TG(+U0(VB@#/E
M;8RQTF`EW;&A]6BB&@U'1*I.\8:4%D*EIT_`YH,LAI:A$.O(H51K#65HG1BK
M;:MS.+CJG%B]TXN3[FGW;$#J.XMXQJR'B+98?B':U\$8K&8*6\ES.:H^!R4K
M\@K=I@A9(O%HID7$D25(AUMB#O)YQ*HP`X1A+A%'`ID8`L$C0*#>V:"OZW]%
MB,^858L(?"$)[&IN7;X[):`8`8R&HJ</,M6O'ENTPWBMQN,-"`B:V62\5U:P
MUK;OBL`6PL@8:K>[9^]M8,P+$;[D-A.X*!G9(O?.+@=V'Z<F)R>W278AU78M
MB1Z?THRD<305>=)[<YG3K9T2P1$K!Q'?;@1&U$B="C:23&10N)*TB9&*]RJP
M?KA"]]HID+FI@QG3%@`1]4=%87^+X#^,]70#HBYW(VOV4@@\Z?#-N5R7RA)V
M-LEC%\[SGQOL3_Y5ZM,8\Y#0H,K9?*8SLKIU.@WM7A,][G=J>+=13QO;1SLG
MN/0JOV#KPZ@AUNLDET-+"!@!W.+(4PBF@&22&4>A6J$*Q!Z[/]<^JJL,5.G>
M!,$<HY-"?R"E"QO_SYK]EXH6B/J7R_Y[^_A(V)8VR+D!_D_Z<G4'@!E22-(&
MV:,V==$8?.*'-^AT`>?.S)L(9##BES"*Y,(9JK.)-&?!=.J@TL7Z$=55:E='
M80NI[VCYI$4M9LM^*&]P>0)-TYS#F@0IW<S:'H],(4E1GBY!,0Z6!9L*9E;R
MKG#2+I%!H?;%I-,OX'@LM#&#HTQW)]B,]3%R2O/&"%I:7]HEM>=L$1%"`TN?
M+[Q;+(?60#R9R12HE-($_'R^]*=(^E$R$.VQ?C`HFG1\\4,.MAW@E8+_"0%K
M22\:S,1\GR#8LC`J"J/9;+'N>]^DV%\XLG1"P@QZL&[(OXLW#"R>J4]^-_@R
MF'O,_A;12Y7TLH8*)U_`.Q++$<I;$F91GY-%+;/X/;I%,F=R6(QQ5,4$GV32
MV"\O.V+-DF47XYR:1JH-YF<I"66MO<<1)#(.TF_M.S&?9<%QHK4M7P8ORY2"
M3DG!9G)"[@<FE%SE#KX71!<!ZN/G`)&//-..HO5BJ7P>S"X7B-1E?-)OQP_$
M30UY,.S4J[4,ZO/M(Z.I>6C9,M?ENYXD5"$BZ`K:6Y9..*Q5#PG14<+N#1^V
M!`-"MC50R'0.^Q*QD4E;+D5I<M<9.3,+-?)\F[`";E'L)7\NT#TSQ<:$="3,
M'IC7-5?VXVFS%F:6$<%7<10I9SG-[_X^@[PVT]?]!JTW3F9DT[]E[47!3=)8
M#;D!I"&%N-,V!29J'Z3$Y5=6^F&QTXK1YM)]'O+/ED/L3.*2!+U$0W4[(GZ)
M:Y%6:US?K^ZW]EOE<KT]\NJ-^EZ]EKP6D;FB^Q#YYNDV.#[@<JC3HG:EF5J6
MRZUJI3FN#]V*EQJUB\O2`G558Y&Z4`HRXG3)%X\&,I+?TU2/!XFO,KY\[+50
M-$8?$LI;70-KJ%55ED?T?;$:$SH]ZDNZ2FRM$+0M0RTGXXSQ&"9L+(8K:%$Y
Q#F6?VZYWL0C96[Z5(^\QGB_V=BK>>5R-.UB=ZE"<*ZCF5?;"J<<)KOTW;/=
H3_;A82EK=+A\4GG[>7K7.F\9I6NK]UASBKO1D[QI9*P#A<&E[#*2T)K7)H@
2U<B]]XHV?7RDY[)G6B_;F<CRL^3"OG#Y6NVB#?+$N7/?2_Z\>EVBLV=.I_@
Y=_W8LX-\U5(=H+\A[10J;6=2L5Q]M'(N.UX3<^M#&,XX"*#<&[@'^2ZTZ`0
;0V6$\TP"B)4UTGO[.J#T-:G8:YJ,9!QR>"M+]#DN8TG*DIZF@H=.%4!_X)>
+)B*K\1'0ELFM=L::DET`@E\%US#%AM`R2L&0GL)%``,Z^-1W2/9#D(Q\R8.
^@[K[P3`MI*;&WLT'-7&?C5Y,&Z3?1?%I5OP1B+APA[/\OP=`VKQT4(I87&Y
PK$[_X+?S/FM-&FBE]?$1W$P>]B&(*V*<H&[7];R=)YK7POJD`3!`-E_,1[P
_P,EO1)O@\*,4&O152?=W#ICO($1@>DL-5Z64KY9>8IV9U.]2"?FGFL8FD5]
PUKTU#?.9$E?@!G@KDV"H0-2S6KIW2O5%?':.]5FM?)-0PR]C,);/%B(_XS,
'L'DR`YE&8.R7D%GBW:T%\_2;4P38T#=$E9JML0N,LS5%A[&6A,)$\9L6B&)
(GBHVFU:C<V]IFF:^J5CA8_KP4S@20PMGD]6U]>PT)`.J"9]9NVUX(PRQR`R
>-QF052A-N$.7,WP*CN^E666-7=+*@W4S3)-DI94/Q:M'&IQ<I'NQ@R3)P,L
BW`)L1),7,&4XJW7,C9Q-(,ZS5%KW4A#86*9\C,7D#?1)&"@)GBII\;I9A&L
KF]885DNEXTATB>5N'`_L+U;V.9YO?'AQ//F><,^5%3(U+#T`P=1=U702E7)
XZ0I2K9^A2]&*_@6C]$9<WDE,HW6UK2BVZP&J;;WV\5:+7M%9R_G:$$[RV#J
C^SI$'F7_(O$7F=H/'.WF2-*9@U,"Y19PC,JA[<(!@6A<7%FB#Y#Z=60RPR/
YV#WW-3]=/"TX];O.)K7L%#Z8?F/&YNVGY$X^5ER!GJJS*G/WD^6=D#C@G2A
58O@P;Z%C"*SW/O)O2;<*AI5<BRH[K,V7L:_Q8E8>-/@UK/#AUG,AZ(@;SE@
IJ?S8($^8,/5>(SAZ/`\1W!</.8<"V/@&(Z*[]%T:/"NW^T<V6>=TRY=CUO5
EG9?]U?HSH"7SC;#VL5V&QDAZ%M2#0!%@*/>Q]LKCD6^_>`]2T`4J>WY*,9-
:]`A4!VC.8J-$'MS\\;K12,[*F,(!S-JO^<JV5]@;F$"YKL)A%)$PF.>N%:M
[A?KE3B1HA%Q5]/I@Q#14@<EXR$3+S$F__>\=]9Y<X*AP^79J]>#$B)N$IL8
9G]$ZB!!"D?+^^@*E08>L;8:$2Z/Y(F%DCS$^S$TC%U++X`.,"!Z8I\JD"-U
8Z)1)1FU@",_80D^B[#DU$9ZM;BOA`JR:[ILR)*8*UW-L1"MH7HINME:]$:3
5C4R4M"'A2$L$5Q#-[*)\#%35U9$5#*)CE;?@4'2#714\JR)62,%L\D#:M)X
EI:!R5CHV36^(F5?QGI,U]W1BGUT_WYI/&O>*GMUWBK-6K':4"3ND1H5X13B
A8K;'7W1!![X^KB<5.78!M%IBLLNN9+7H#/F9$):L>N7JYH!]/$9NS)L5\;.
2=W)15I1_'^A4+&XZ9('<4)#Y$#3*G&!1UH%!B1U.`H;FS2@'"1`&*QP[H!L
9/G+T)N,BQI!VK8P_H0E+.D%[A$LPKA63KDD&DZ(=:^]YWH8+[]<=EMNHS9N
#-V8,XSA=J@Y&M;J58[CRO_&Z.K"F=I#8C2A"_E^YU2HOC$@";TO6EVB,-O;
```

```
M,&'S(L=]#X'X>6XAQN9A61Q7#8$%\JH($%$#9`RU%Z@%F'BSZ^5-4C1SW%L_
M]#;,7F3B?MHY>F^_NWK;O>B\[<9<E^C;T?G9`#DJO/<`;B&@P%-L8_J7]Z<4
M66H8>B@:!&/$8QR!M#!##X'3TRN*A_`[M5&V(\V'[7<I5?4\FEG6_=`$?]F<
M?,8CTEB>B`NY7`UIA:J_52#\=L7Q1GN>4R[7:PVO4A^-O)BG;)2'UVGTFXST
MVQP3'_ZIUHS[6MAL(.W,;;[`>VU=7@ZZ%W:73O+?^,?9>:__5_7WH'?:[1^P
MM1J5\?;HC:"&$;:ENM4C$\PDYB4#'\@37X*PF=GQ9!:G%,N#O$,@%="GT+Z?
M3@ZT8+$);.E81`32\M'N54-S0X$1TCY(_?Q>O57?;^UCC!6G477'C5JMV3;'
M/:,`GH2,CVPM1*J0E@`IP*L4[`:1CCR.)#I>CIS)Q*:+!V_IV:-A(=5'=Y,G
M#F4<QY1=+`J/39<PZ&4'#&R@&\@VOJ)W_V<T?&6-I=TJ^_1,(E\$H8>FEWAA
M+_OEFE-%3C8B/%'Y!A9__(U4$K=J56^_WA@AG,=XW&Z.O%;3,R<GGI-G)?Z6
MI1N:COTB!0.#I?ZAMX78-JCOLFZ\R9R,ZO#@W+*Z_'5KYZA[G,_)JQ20*-]U
M+NU._VTQ%2RA*/T-<B5E,$V!/\E[%'8BQN'CZ-W"NH:^$O86E=?I'[ZS.W"Z
MXCQ0U;=\K04)*F:=XH-8'[F2^"T7@\!,L.1K?X9B&!O]R&ZFU2O&)0N\YM3Y
MY&&WRL'P[Z&)-&-\$A/8'#GC4=4;MD;E<JWF[;?;L-'J,3R6K"(26#;F9Z9Y
M3/+X:(;7I0<,9:6L;\N!!5P-SJVW*`=;48K1XF&^#-*^^'?3Z?U2?8$!'$4_
M9AZ(92F9IM'+'?62H7<")&#X[C_R$D;D_'@`!V.!TH0CGY)D#;@H=V2.A'PK
M#X]V8U2K._M#&.:FTQZYU6JKV<H<9I4[,<+J"PUNC>_;$Q@K1.J0T9^XX9(A
M5A@.WKAL$<A#>-T2N0SU3\E;Z+C?_:MP0R1\5]0?]#Z<=E]95>OMNU^+6I1Z
M>=U%TH=P0!)W=XGS@`*0U=M5;/<>'H3UMN+OH7LD'0%%4U9V!C110=>!"_,M
M<8X+26S7>D=,,Y$ZNOS7G:\B.[Y1J&&,V3Q%:/;[X?#"?O.7BX$6Y@X?#5U)
M@%?%=+&[EKCS05]QW+S</HW99XPM7%"->'BW-?C2>&T]'[V.\&YE04E\OKC8
M:XZ%[H(&W"[B<'%TY<W&Y*+?/>XBVE/GS7E_8/WVFZZN6Y<1XZN+3+$Q%5%S
M^=E&"9D-;=$$!Y?O:BE<EH%.%RV'5L@81"#2`XA3C$TG,7BU418,:"C,KRE2
MN+44,)T<94%/:^1#4TLR:5R1]1M(@SQ&5*"><C<FQ7%,!4/W:>!KR463@48-
M>^MK41J-MGPV5B3%BA`C96A[OS6@1%;_Y?*3]5QZ4Q^Y<'8WGCHSX%-#D`WQ
M4LT*F!V1>GZ82A>#'),LNA5-B.NQK>'(6:%Q"6912ZU(EB*:[*J^6!/OUIL4
M53D@#U'$\&%DW84+H?P(`ER,I(N7FR+`#>MMKUT?UNK[FR'`C73#DGJC+BQ+
M)+'&6=L-0AB^F,.NR=8F/^'_2L#NP3=<%K"6$TDX9$`L\\W=;AC-H/H[EBHZ
M0Z)WA#*X2_\7E_EI%_Q;.SK0G`F[*&Q!=$E>(*83IA_P!3@!`@T9MFO1.C0@
M[!0DLC`N(\9J`[1*3'R'-F5%Z^8.E1ZD^>!"_-">WSR$/K#.)O8RD">TM[4Y
GU]DNP`?%A6T!"1.+>K.GKTDI6JU]5%[2X8#MXJ"0!80`:D9M`>@&YH).K\W
XOY:4?V0<2\EK*^9HMH22:*&BLL;H:V\PSCE>1XU1:TXHFKP4L,;,_(?]=[;
_?.KLR/[ZB(O6[]7M.KZ6:6CBYOHV%S\?6;Q49$%:]<X1/4RN9352Z-0]PO;
#/]5@9>O9[9<';+BM@YG[`<-#D'.E;ED](4F<2YI[0Y`7.D.[./3@3V?W%NY
5W#.X[HSM(7&;*XK0I9@.LK$R\-@U-':E`V>0#?42HDUF7+(;P?9'8V?;AW6
'5ZBZM#:=G#8D6-PP@,SW:DW'=SWO1`5A^BL)'2.-ND<V=?""7E@-MC)L>>T
>SKXT!D,^I?VU=GE1?>P=]SK'HG[ATD<4)LL4%'S0=D(E/J?R2J-"<A9A\Z,
X,$I8K(@5RF@X/CH:TD^VN%M!J/7&X4$4A#"Q9WM>L/5-3$73*6X,WA#;7T?
6XW_^C9K"R)V$RPVB0^-2JX.06MH[R4^FA1"/K2IJRFPAVBX/'%70WN>AZ&P
=BQ_TXY0D;7L(N^^LLA&1I%YY5<Y<2=?6?9>9G/_\>4E?C9_QI9)8D&DGQ!&
\\R3(KW@Y]]MYZR+?B]H-<Q3\_:+1L(\-=*K`@FJPI7=?V-EJPWZQ36MOK$F
=^.:W*^N*3;Q/DJH=S$*KZ4Q:X^1"F*8(/]$/V?@:"Q%KSY+-;+!VZ6A1*_'
)"Y:BO<SF3'_(/K[+L81,5?(0+FOK5HE^=H>/BP].J$DGJX:#$Z+UA)Y)G`6
$K1Z[<#:V?&3)S_!-R5%/7/\+B]>H6A?J;9TZ9XAQ_V/^M"*(R$J.DG>S:)/
^E]8]-K2/CROU-Q8@?ZZ,@U)U"SK0C`*KV*K)XW!IQPX_S7BA&E.BKC?BY)-
-YI@E<R)9'";[*:\YP!_\99`NG])8RIZ)9D[*;%%C#L"OHU/,Z"WTO>)><.@
FPVH?2,M).:P%9_#-,?KH.F&0N`[7M);X_%D%=[DPZ6+V\\T3(B<!6)&9H:F
A8.>B!AC*@2+HT=4*^,5^@WB8CI1H#BC$.!:?!<C@B"NSG+A7U^#3.\F6L]H
F67+ZM[S];112!2##I5)J/FAB_OR.G5/19NJK(LOUL%)RQ`M`A_Y>3KT!_:>
!I9]:E#GY"\CHY%'K5%&(7[X+A(-U"BGU.MHMCME$?K.$R.B?5(%H/Y,6-63
`C&06<D!6!D+DK<QFO_(@'C<-J,8(_`,9Q`V]6;=PAF64LB("ZH@H1,,9F7U
ZJS\IDR1.\-P-24L.#*S81L&#*<J4VH#&@OI$W.SD:G2C!NK!](YA=.)^]**
NAG=0?NH8/[`UN(4R6F^7/Q<*9?#7S_"N'6HF9+-MA;.[!I!$2V\GO88SC!U
R^-?6.2C1Z+I,;E<Q-Z$OT9;/GX3&?XJ>L_'ZG@>V9"BA@,/O'N\#4C194PB
`S%A_Z8%7GH=$;TQ#N(XF'NS?'0;EUL,<QH%>38V;5.E&8W,8'`U8@)4J(C/
:E["7PU]!IZ:8Q(@I814+5I84Z;93EI]*=69Y!,*?V4]G]P#L=S.T\#BV&T7
X'WA((-F9HMT-'^JO4F)3IX?4"*T>4-A+K,/X]$D"+W\6(%Q15&M:6E\7G\>
92Q/PZLTOAHQ8^)XTNP1$G%#I--;M)6UL"`>&G::^31`GOA>3>N)&\P\[HCQ
^E97]Z4M<,5BT-F`R0E.E6BK=@HQ]<,]+J@%+@[M8FG]YHZ:(+=>^#!S%\%4
Q?)0RN^R^O("-NWQ\<&7[V,6[ZR\+,E8>'185F*2"86"25@L%%)U$(]=>I&8
(@*20ID9^HC;*1'F?/_JS$9HTJY]U'US]3:>-";?Q$4@H4NP8KGDN"!NQVN+
K0/SA6A`S=2PU=!9@`PTM@N\:5]V7J8IELC3&8HD,IBC.X$5%P]4X]SN'R'S
D%1'Y3$3$:N4\92FAF+_8]%\V\`%9XQ?G!1D#)CAC"VT91G76'I;HN@DOUM+
!(5R"XGQ7T=&-99<V1NG$U3Y?!5AW:`O,?G*N.\HRQ;2!MVK[_&E^V_0+UB!
WW]O56OQVE(+(-<'H\N)P8IBTM]77HF+Y:5U7WT573+'NY71ULW253_&DTE/
7_V=T6;D$]*K/,C,5<W,58WG2J4!<>7?[Q=U/[/*>JQ*L782$IY8QK&U7(@7
%]<U2GL^R3VF%U,T!S%1:G-=([^AW%9ZN=$AG-G9=*5P@@>4YWU<FD82C"32
)KL!V@G,\R9U273UYBTT=39Y1;C^@EQ1"$"62O'"W9R>8K[PQOX])R`+`.,K
,I940+6V)]>F"&"!;;0T;R2D;(2XG1>5%JT7X5(1X-]^LYY=VKW+HUX?4I7#
M)4/OF9+WKB7:B`TGB&VZ-P?^)%R:2JOI)T@1U83WVLD3',M3N!7LHZ17D$5L
M\&#_18W6+SE,ZXV6*`@1;;UU_`F*8N6R-1#6&L`"I%!9><)@@=3@1(JOM:5(
M79`&)?\<XYHQ;@R&&0SNT-D`L1"EX3W-Z4R<)'+"5203^:*`&MSP>5AZ/G%+
MS]TRC!2P`6H">!FE7'9!A46Y.F5MXI9%G%):G<=T?_6,8AR**SV=F^-2=G:T
M;GYKPY,-IHGCWJ0V7L?BL?*X$Y6L&+4A=[>3XVB-.,HI?3%7VZ%:I<01,<8]
MAN'%$E_!8GP>_I(CU9;JE38&^C*CAB1A.;]IB65*8G%111,3V)AH4T$!J=HV
MJ]95D!$>8BL?(X`O5"I%6O3VR9QZ[)+Q="EB3^:B*)X4>/677WJ:L1FU^9D6
M^!-&/'U]\),[[O0M-G@CQJ1[HOU\+"N&:;JOZ-GY554O(M+EBU6N.@\K@7M5
MC(M4M^(R>H.+9V&@MT'*3=FJ#7-&AZS:!U_=G<<Z\:5-7]/@S+F`O:FM+%CG
MEECE^E*272:[4<P;,EX5[`RM*)0QCZZ`1SL\/^I&^JA'JCQ"`Z;0FPXG#^F5
M"CT!&3KIQ2BY8XY@C[4*'*7WC<K&]9ZR?O`(OJ?7FW9=IKBP:"MO/KA=-,@;
M8V`)M'9,U&FHB%+*-TC6>I.ZFS3+MY3P3.DF=35OB$YBX_%H,Y,Z(U93+%33
M10?M68U@3=&K+]7.''PYG:;12P8[DFU(A#M"C#WXCXT2Y=_2NMRKC>KU40/'
MR*W6W'&M-FS$_.>B/&*PU&\&8Z]0Q![X1W,JEY[.>B0&=AJR^-;AT63_E.PM
M\=[D.B1]6]DY4QU)S^B]P6Z2N^SR'M/R7S$1,&[X3*M4E"Y\YBG=ESJA4J;!
MX=M#>0F,.5!=A7CJ><$*;8<JJ1%1WB]B7&ACZFXG-&OXC_(EJ[6&]5JUWBB7
MVVZM6O.:U7$,.IJ2\US1GV3PSPYD]83!/_`B:!\:>B-TZ&&+__A'G&U"=GDL
M%EG,F^ORZ(1>":1F-V_;G8N+DZYMDP0BW^*E5D$'GH%\"#K#$$V\ON!?`?V<
M!3H.U`>1N-4/V"KS#,]WY4";^/(Y^4I*X#&=$HJB:7Y"*887&L!"GAM7".9(
M<E+40#+NX-JZA`/0JV1CY\XB].3W/->2UD]1S;\9@?3I^7<^Z_!_*:`$?=G]
MICH(Y;?=S,#_I0?Q?RO-1AV^0+IJ&TC;GZSF[]3'M<__<OS?#><?#^6I5YZZ
M7U/'>OSG2K7:JFGXS[4_56KP_R?\YS_D^<[J_^7"HIE6&$8"Z'%KZX(7`L%<
MN6XL5;B:$[\O82$EK"&6-UXX4P_UL&4*&^1Z2\>?A%L+;\Q8AYC.60!/-_'*
M6UOH]@G%+2CB/#I=H7<Q_L?./MYLY'L8,?8?*W_!00D(*&F^"!#$K&@YH871
M+(F#VH*B24\YBN!^H`KA)?O*(G#RK:WOOK.N0N?:V]H:!!:%[GRUM?5?__5?
M6_]A(=,%L@IP]/#7J]U=^'^9O(U%?>5@<4WNQV7X`NE''%I$YKSQ1I]0`77+
M-?%;9XZQGZ/]Y"T8&AT^Z_$^8]$Y'6<QNFDU2F$P7DZI!A6`$YNZ=15ZP@L*
K=YQ3F@BAOX,(S0A$MB4L,+8N"::$NBR#/9RXVU)9V]GXB\?S*3L]L1!<]'K
":?87`0.>F2%HX4_I-''T!Y;QMB3!8\QW?_N%?_TZ,^&]#^Y=+^@CO7TO];&
,]^D_[56H_U$__^(QPQZ/1IY$Q+M,=B/"K:3]EK&'1_M-4=USZMZY?)^I>+M
-4:.$\>A3LLN(F*G?>*@MR).'?R;C,E(49PH="[&67*F8?[R838Z1$NX$,1M
E`5)-HW'>#(!+E`K8(("H,'>T,9FQ&/,)Z/?^\MA%*<J+09^#$5'R/H@KL)O
>XG:!ZDJ2)0]<1#$8*@!U5"S?)E#VI<@0A=#1A*@Y7)4%O'UE5I$.HXKQ/FC
-YV&0)R'K6_/@XD_>K!O_8`;D'4;<G%^TCO\R7K?.S\A."[$H+]D+^H^7F_'
@ZG]/O??VKU.2E<ZC<-#T97;Z116PLP?9;7__>FI==$YZQUBPW_T9/@!O-X9
LELW?/B7]<%H$-DMV7`L8LBAA3)@ZG[H'A8M`4RZ/BA,#E;,R"-;Z^=SZ^?<
VL0QU[[=7.+-\^_N/UK/0S+=>`IG]$]#*?T4SN@IG-%3?)S_5O%Q]IMMISJN
C\;?'!^G1H-;DQ#D6C`%9^[OXO]*$EZM)"X`A/P6TZ!3>G8,B7U`Q3K%D"RQ
C6-6>(0821D<OC70V!=3+51/=AB?6/B<F@C@_M\\?,Y3<)QX<)PO#"#S[>8H
FX2.^9W9J:<(,=7V7KU6:]?&PZ^($%-K5?\G18AY-!;,4S28IV@P3]%@Q/,4
#>8I&LQ3-!AZGJ+!/$6#,:MZB@;S%`W&>HH&\Q0-YG]6-)@_V%]<HC:L4<#(
C*^MYRXM`N4OKNVE)Y_SW\GG7$\+DQ.)BLYDY6DYDDFUF[2=CR1=H\&MR,#+
<K84@&"OXNHTW073+/G)[3W1EB>W][5]09*[H=][Y7[_3;KC>U8AU8\ZI8F=
7T_>]OJ[_V7>]D\>XT\>XT\>XT\>XT\>XT\>X[&L3Q[CE/7)8_S)8_S)8_R_
O\?XO]L8_>EY>IZ>I^?I>7J>GJ?GZ7EZGIZGY^EY>IZ>I^?I>7J>GJ?GZ7EZ
GGIZGY^EY>IZ>I^?I>7J>GJ?GZ7EZGIZGY^G9^/G_`!'U1^<`6`<`
`
end
```

|=[EOF]=---=|