I • Ternary Machine

A ternary numerical system (also called base 3) has three as its base. Just as a binary digit is called a *bit* in base 2, a ternary digit is surprisingly called a *trit*. The *trits* are 0, 1 and 2. For this problem, you will write an **interpreter** for a ternary machine that accepts as input a program consisting of a string of *trits* and executes it. Instructions for this machine consist of an opcode sometimes followed by a parameter. The opcodes for the machine follow. **S1** and **S2** refer to the item at the top of the stack and the 2nd item on the stack respectively. When these are used below, it implies they are removed from the stack for the given instruction (popped).

Opcode	Parameter	Description
000	***Number***	Push positive ***Number*** onto stack
001	***Number***	Push negative ***Number*** onto stack
020		Duplicate the top item on the stack
021		Swap the top two items on the stack
022		Discard the top item on the stack
1000		Add **S1** + **S2** and push result onto stack.
1001		Subtract **S2** – **S1** and push result onto stack.
1002		Multiply **S1** x **S2** and push result onto stack.
1010		Divide **S2** ÷**S1** and push result onto stack.
1011		Compute module **S2** % **S1** and push result onto stack.
110		Store the *Value* **S1** at the machine's memory heap address specified by **S2**.
111		Push the *Value* from the machine's memory heap address specified by **S1** onto the stack.
200	***Label***	Set a label *mark* at the point in the program after this instruction. This may the destination of a *Jump* or *Call*.
201	***Label***	Call the subroutine at the specified label.
202	***Label***	Jump to the specified label.
210	***Label***	Jump to the label if **S1** is 0.
211	***Label***	Jump to the label **S1** is negative.
212		End subroutine and transfer control back to the location where it was called from.
222		End the program and halt the machine.
1200		Output the character from **S1**.
1201		Output the decimal number from **S1**.
1210		Read a character from the input and place it in the machine's memory heap at the address specified by **S1**.
1211		Read a decimal number from the input and place it in the machine's memory heap at the address specified by **S1**.

The machine you will implement must support a stack of precisely 1024 items. Items can only be ***Value***s or characters. The memory heap is of arbitrary size, and possibly sparse. Addresses used to reference the heap are positive ***Value***s.

Number is a decimal integer from 1 to 31 bits in length. (Yes, it's in **bits** not **trits**.) The end of the value is indicated by the *trit* 2. Ex. `100112` = `19` (decimal).

Value is a signed ***Number***. Since ***Number*** is no more than 31 bits, a ***Value*** can be represented using a 32-bit quantity (`-2147483647` <= ***Value*** <= `2147483647`).

Label is a string of 1 to 128 *trits* of value 0 or 1. Its end is always indicated by a *trit* of value 2. If an attempt is made to redefine a ***Label*** mark, that is a RUN-TIME ERROR.

Subroutines calls may be nested (and recursive) to a maximum depth of 1024 (the top-level execution depth is 0).

Any illegal character (non-*trit*) encountered during execution should cause a RUN-TIME ERROR.

Program execution starts with the first character in the program and continues until the opcode **`222`** is reached or there is a RUN-TIME ERROR.

Input

Input consists of one or more lines. The first line contains a string of up to 8192 characters representing the program to execute (the terminating line feed is not part of the program to execute). If the program requires input (for opcodes **`1210`** and **`1211`**), the input follows after the first line (after the line feed), on one or more lines as required by the program. If a program requires input, there will always be enough input supplied. Care should be taken when implementing the **`1210`** and **`1211`** opcodes so as to not consume too much input. Ex. For opcode **`1211`**, only an optional leading minus sign and decimal digits should be read until a non-decimal digit is reached.

Output

The output consists of at least one line which is the output of the supplied ternary program and/or the message "RUN-TIME ERROR" if an error occurred during execution of the ternary program. Some possible RUN-TIME ERRORs include invalid character, invalid opcode, stack underflow, stack overflow, division by 0, undefined label, etc. If a RUN-TIME ERROR does occur, be sure to flush all pending output from the ternary program prior to printing RUN-TIME ERROR. Attempting to access an address in the machine's heap that has not been previously set should retrieve a value of 0 – this is not a RUN-TIME error.

Note: It is possible that the ternary program will generate some output and then get a RUN-TIME ERROR. It is also possible that there are potential errors in the ternary program but these errors will not be reached during execution (ex. invalid character/opcode, missing label, redefined label, etc.). In this case, these RUN-TIME ERRORs will not be reported (which is correct).

(Sample input and output is on the next page)

Sample 1:

Sample Input	Sample Output
000122000100001120201201000101021200000121000020000 1 011210012100100010122020100001122000100010120222222	1 2 3 4 5 6 7 8 9 10

Note: There is no *newline* after the first row of sample input. The input in this case is one long line beginning with `00012` and ending with `22222`. It is just split above so it fits in the table. There is a *newline* after the `22222`.

Sample 2:

Sample Input	Sample Output
000100100021200000110111121200000111011121200000 010000021200000110110121200000110000121200000011 011102120000011110012120000011111121200000010000 021200000102121100002000120201201000101021200200 012020000120211101000000121110210201201000101020 120000010211100012100102000010202111021110220212 2200102222 12	How many? 1 1 2 3 5 8 13 21 34 55 89 144 233 377

Note: There are no *newlines* at the end of rows 1 through 6 of the sample input. The input in this case is one long line beginning with `00010` and ending with `02222`. It is just split above so it fits in the table. There is a *newline* after the `02222` on the 7th row. In addition, the last line of input, "`12`", is the input to the ternary program. In this case, the program requires a single number as input in response to the "`How many?`" question.

Sample 3:

Sample Input	Sample Output
00010000012120020211112These are Illegal: RUN-TIME Error200111200010102120022 2	ARUN-TIME ERROR

Note: There is no *newline* after the first row of sample input. The input in this case is one long line beginning with `00010` and ending with `00222`. It is just split above so it fits in the table. There is a *newline* after the `00222`. Read the **Sample Output** very closely.

Sample 4:

Sample Input	Sample Output
00010000012120020211122021002200111200010102120022This is NOT an error!!222	A

Note: There is no *newline* after the first row of sample input. The input in this case is one long line beginning with `00010` and ending with `!!222`. It is just split above so it fits in the table. There is a *newline* after the `!!222`. In the **Sample Output** there is a *newline* after the "`A`".