BrainF__k interpreter

HackerRank

BrainF__k is an esoteric programming languages. It is designed to provide a tongue-twister to programmers, where even adding two numbers can be more difficult than writing a complex algorithm in imperative languages.

For this problem, a BrainF__k program is allocated a continuous memory of infinite bytes, where memory locations are indexed from 0 onwards.

Following are the commands used in this language:

```
>   Increment data pointer so that it points to next location in memory.

<   Decrement data pointer so that it points to previous locaion in memory.

+   Increment the byte pointed by data pointer by 1. If it is already at its maximum value, 255, then new
value will be 0.

-   Decrement the byte pointed by data pointer by 1. If it is at its minimum value, 0, then new value will be
255.

.   Output the character represented by the byte at the data pointer.

,   Read one byte and store it at the memory location pointed by data pointer.

[   If the byte pointed by data pointer is zero, then move instruction pointer to next matching ']',
otherwise move instruction pointer to next command.

]   If the byte pointed by data pointer is non-zero, then move instruction pointer to previous matching '['
command, otherwise to next command.
```

Each of the above command represents a single operation.

Objective:
Given a valid BrainF__k program and an input string, you have to print the result of the program when executed. All those characters of the program which does not represent a valid command can be considered as comment and should be ignored.

You have to print the output for first 10^5 operations. If program executes more than 10^5 operations then you have stop execution and print "`PROCESS TIME OUT. KILLED!!!`" (without quotes) in the next line.

NOTE:

1. Initally all memory locations contain 0. A location can store integer in range [0 .. 255].
2. At the start of program, data pointer is at memory location 0. It is guaranteed that data pointer will never point to a negative memory index during the execution of program.
3. Number of read operations will not exceed input string length.
4. Program will not have a mis-matched bracket (`[` or `]`).

Input
First line will contain two space separated integers, `n m`, which represent number of characters in input

to BrainF__k program and number of lines in the program, respectively. Next line contains `n+1` characters which represents the input for the BrainF__k program. This line ends with character '`$`' which represent the end of input. Please ignore this in input. Then follows `m` lines which is the BrainF__k program.

Output

You have to print the output of program as mentioned in *Objective*. For programs with more than 100000 operations, print the output till then followed by "`PROCESS TIME OUT. KILLED!!!`" in the next line.

Constraints

0 <= n <= 150
1 <= m <= 150
Length of Brain__k program will not exceed 5000.

Sample Input #00

```
0 20
$
+++++ +++++             initialize counter (cell #0) to 10
[                       use loop to set the next four cells to 70/100/30/10
    > +++++ ++              add  7 to cell #1
    > +++++ +++++           add 10 to cell #2
    > +++                   add  3 to cell #3
    > +                     add  1 to cell #4
    <<<< -                  decrement counter (cell #0)
]
> ++ .                  print 'H'
> + .                   print 'e'
+++++ ++ .              print 'l'
.                       print 'l'
+++ .                   print 'o'
> ++ .                  print ' '
<< +++++ +++++ +++++ .  print 'W'
> .                     print 'o'
+++ .                   print 'r'
----- - .               print 'l'
----- --- .             print 'd'
> + .                   print '!'
```

Sample Output #00

```
Hello World!
```

Explanation #00

Here `n = 0` means that there's no input to the BrainF__k program. That's why second line only contains `$` which represents the end of input. Then follows `m = 20` lines which represents the complete BrainF__k program.

Sample Input #01

```
6 6
abcxyz$
,+. This program will 6 characters
,+. For first 3 characters it will
,+. print its successor
,-. For last 3 characters it will
```

```
,-. print its predecessor
,-.
```

Sample Output #01

```
bcdwxy
```

Explanation #01

This program six characters, for first three it prints its successor and for rest its predecessor.

Sample Input #02

```
2 10
pm$
++
[           loop will execute only 2 time
    >
    ,       reads a value
    +++     increase by 3
    .       print it
    <
    -
]
+[]
```

Sample Output #02

```
sp
PROCESS TIME OUT. KILLED!!!
```

Explanation #02

Total number of operations executed here is `22` till second last line in program. Then it enters in a infinte loop in next line.