Problem B
BrainFuckVM

Given a brainfuck program, determine whether it terminates or enters an endless loop.

A brainfuck interpreter has a data array (consisting of unsigned 8-bit integers) with an index, the so called "*data index*"; the entry of the array pointed to by the data index is the so called "*current entry*". A brainfuck program consists of a sequence of the following eight instructions:

`-`	decrease *current entry* by 1 (modulo 2^8)
`+`	increase *current entry* by 1 (modulo 2^8)
`<`	decrease *data index* by 1
`>`	increase *data index* by 1
`[`	jump behind the matching `]`, if the *current entry* is equal to 0
`]`	jump behind the matching `[` if the *current entry* is *not* equal to 0
`.`	print the *current entry* as character
`,`	read a character and save it into the *current entry*. On end of input save 255.

Interpretation of a brainfuck program starts at the first instruction, and terminates if the instruction pointer leaves the end of the program. After an instruction is executed, the instruction pointer advances to the instruction to the right (except, of course, if the loop instructions `[` or `]` cause a jump).

In addition to the program, you will be given the size of the data array. The entries of the data array shall be unsigned 8-bit integers, with usual over- or underflow behaviour. At the start of the program, the data array entries and the data index shall be initialized to zero.

Incrementing or decrementing the data index beyond the boundaries of the data array shall make it re-enter the data array at the other boundary; e.g. decrementing the data index when it is zero shall set it to the size of the data array minus one.

The `[` and `]` instructions define loops and are allowed to nest. Every given program will be well-formed, i.e. when traversing the program from left to right, the number of `[` instructions minus the number of `]` instructions will always be greater or equal zero, and at the end of the program it will be equal to zero.

For the purposes of the problem, discard the output of the interpreted program. [1]

Input

The input starts with a line containing the number of test cases t ($0 < t \leq 20$). Each test case consists of three lines. The first line contains three integers s_m, s_c, s_i, describing the size of the memory, the size of the program code and the size of the input ($0 < s_m \leq 100\,000; 0 < s_c, s_i < 4\,096$).
The second line contains the brainfuck program code to be executed; it consists of s_c characters.
The third line contains the input of the program, as text (only printable, non-whitespace characters). Once the program has consumed all available input, the input instruction shall set the current cell to 255.

Output

For each test case, print one line, containing either "`Terminates`" or "`Loops`", depending on whether the program either terminates after a finite number of steps, or enters an endless loop. If the program loops, also print the indices (0-based) of the two `[` and the `]` symbols in the code array that correspond to the endless loop. You may safely assume that after 50 000 000 instructions, a program either terminated or hangs in an endless loop, which then was executed at least once. During each iteration of the endless loop at most 50 000 000 instructions are executed.

[1] For purposes of debugging, you may examine it. The third program from the sample input would print "ICPC"; the fourth program, when given an arbitrary string, will print a brainfuck program, that in turn will print said arbitrary string when executed.

Sample Input

```
4
10 4 3
+-.,
qwe
1000 5 1
+[+-]
a
100 74 4
+++++[->++<]>[-<+++++++>]<[->+>+>+>+<<<<]>+++>--->++++++++++>---<<<.>.>.>.
xxyz
9999 52 14
+++++[>+++++++++<-],+[-[>--.++>+<<-]>+.->[<.>-]<<,+]
this_is_a_test
```

Sample Output

```
Terminates
Loops 1 4
Terminates
Terminates
```