Problem G. Game of Pipes

Time limit:	3 seconds
Memory limit:	1024 megabytes

This is an interactive problem.

The game "Plumber Mariusz" consists of $n+1$ levels (numbered from 0 to n), each of which has k versions (numbered from 0 to $k-1$). Each version of every level (except the last one) ends with two pipes – the left and the right – and Mariusz must jump into one of them. Each pipe contains some number of coins and leads to one of the versions of the next level. **Exactly two pipes lead to each version of each level (except level zero)**; these may be any two among the $2 \cdot k$ pipes coming out of all k versions of the previous level.

A single playthrough consists of choosing a version of level zero and then choosing left or right pipe n times, and the *result* of the playthrough is the total number of coins collected in the n visited pipes. We do not learn how many coins were in each pipe or which versions of the levels were visited. Coins in the pipes always reset to their initial values, so all playthroughs are independent. A playthrough can therefore be represented by the function:

`int rozgrywka(int ver, string s)`, where $ver \in [0, k-1]$ and s is a string of n letters `L` and `P`.

Note that `L` and `P` correspond to the polish words for *left* and *right* respectively.

You may perform up to 300 000 test playthroughs and learn their results. After that, you will receive up to 10 000 queries about playthroughs. Your task is to determine their results.

Each test is fixed before the interaction begins (i.e. the interactor is not adaptive). A test contains a directed weighted graph with $(n+1) \cdot k$ vertices arranged in $n+1$ layers (levels), and q queries (ver, s). Each vertex (except those in the last layer) has two outgoing edges (left and right), each described by a coin count $c \in [1, 10^8]$ and a version $v \in [0, k-1]$ it leads to. Recall that every vertex except those in layer zero has in-degree exactly 2.

Buffer flushing

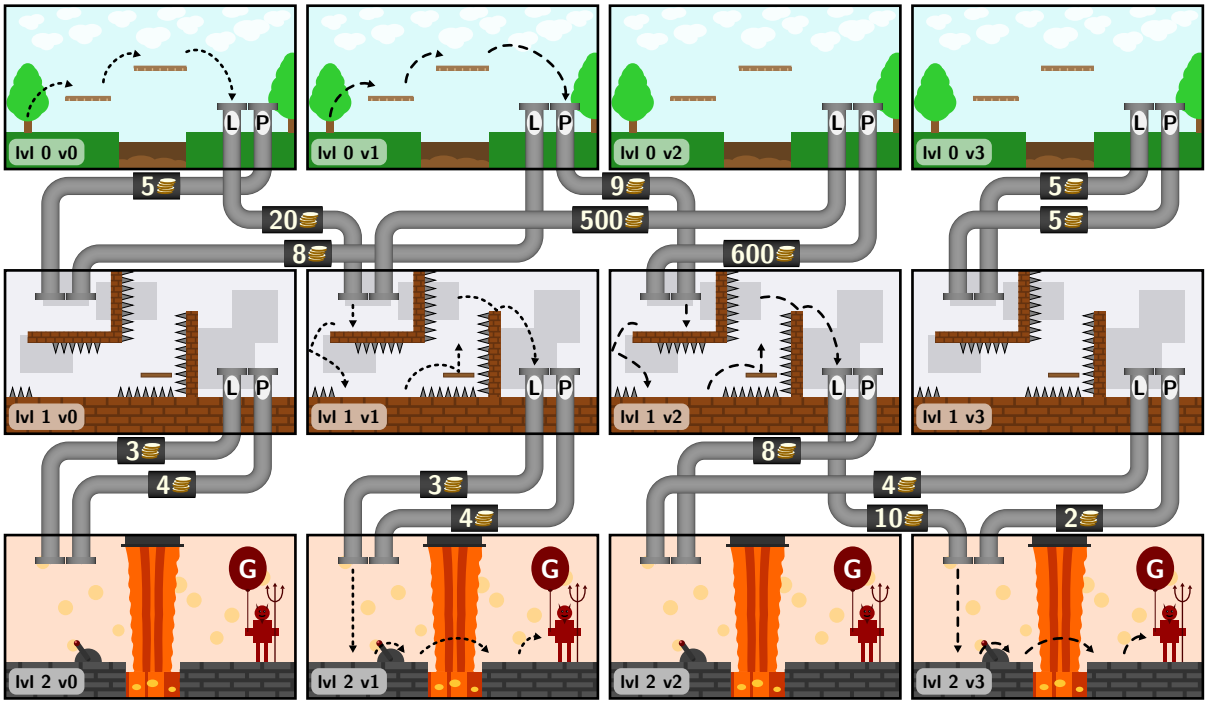
After outputting each line you must flush the output buffer, so that the interactor receives the output line. For example, in C++ you may output a line containing an exclamation mark using `cout << "!" << endl;` or using `printf("!\n"); fflush(stdout);`. In Python you may use `print("!", flush=True)`. We do not guarantee stable behavior of the interactor if you output multiple lines without flushing after each of them, or if you output many lines without reading the interactor's responses. Your program may not open any files.

Interaction Protocol

The full communication flow is described below:

- Read the first line containing two integers n and k ($1 \le n, k \le 20$) – there are $n+1$ levels, each with k versions.
- Up to 300 000 times, perform a test playthrough:
  - Output a line of the form „`? ver s`" (without quotes), where $ver \in [0, k-1]$ and s is a string of n letters `L` and `P`. **Flush**.
  - Read the next line containing the result of the playthrough – an integer x ($n \le x \le n \cdot 10^8 \le 2 \cdot 10^9$).
- Output a line containing a single character `!` (exclamation mark). **Flush**.
- Read the next line containing an integer q ($1 \le q \le 10\,000$) – the number of queries.

- For each of the q queries:
  - Read the next line containing an integer ver and a string s (same limits as above).
  - Output a line containing the result of the playthrough (an integer). **Flush**.



The illustration shows the first sample test ($n = 2$; $k = 4$). Two final queries are marked using dashed lines: `rozgrywka(0, "LL")` $= 20 + 3 = 23$ and `rozgrywka(1, "PL")` $= 9 + 10 = 19$.

Example

Here is an example communication flow:

Your program	Interactor	Description
	`2 4`	3 levels ($0 \dots 2$), each with 4 versions ($0 \dots 3$).
`? 0 LL`		
	`23`	First dashed path in the figure. Start from version 0 of level 0, jump into the left pipe (20 coins). From version 1 of level 1 again jump into the left pipe (3 coins). Result: $20+3 = 23$.
`? 0 LL`		
	`23`	Same playthrough, same result.
`? 2 PL`		
	`610`	$600 + 10 = 610$.
`? 0 PP`		
	`9`	$5 + 4 = 9$.
`? 3 LP`		
	`7`	$5 + 2 = 7$.
`!`		End of test playthroughs.
	`2`	There are $q = 2$ queries to answer.
	`0 LL`	Lucky – we tested this exact playthrough!
`23`		
	`1 PL`	We do not know the result and cannot deduce it. We never tested starting from version 1 of level 0.
`19`		By chance, we guess the correct result: $9 + 10 = 19$.

Note

In the attachment (you can download it in the "attachment" tab at QOJ) you will find sample (incorrect) solutions `G.cpp` and `G.py` that perform the communication correctly, but compute the results incorrectly. The directory also includes: a sample checker `Gsoc.cpp` and sample tests (`G0a.in`, `G0b.in`, `G0c.in`, and a large random test `G0d.in` with $n = k = 20$). The format of test files is described in the comment at the beginning of `Gsoc.cpp`.

The sample checker `Gsoc.cpp` differs from the one used in QOJ. It might not validate input or the arguments of the playthrough function.

You can run your solution using the script `run.sh`. The command takes the compiled program and the test file name. For example, to run the sample programs on the sample test `G0a.in`:

- C++: `g++ G.cpp -o G.e && ./run.sh "./G.e" G0a.in`
- Python: `./run.sh "python3 G.py" G0a.in`