Chess Rush

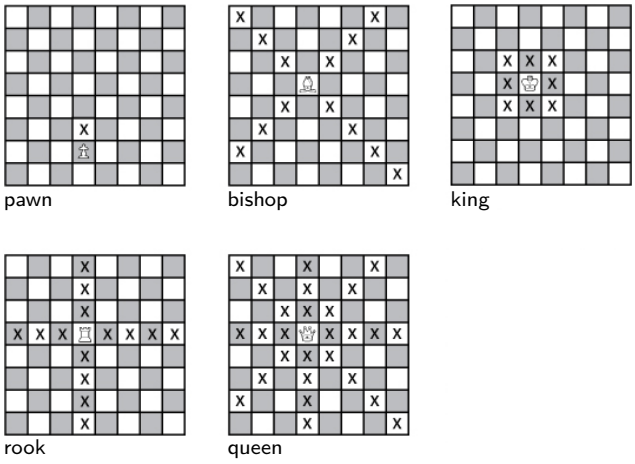
The mythic world of Chess Land is a rectangular grid of squares with R rows and C columns, R being greater than or equal to C. Its rows and columns are numbered from 1 to R and 1 to C, respectively.

The inhabitants of Chess Land are usually mentioned as *pieces* in everyday language, and there are 5 specific types of them roaming the land: pawns, rooks, bishops, queens and kings. Contrary to popular belief, chivalry is long dead in Chess Land, so there are no knights to be found.

Each piece is unique in the way it moves around from square to square: in one step,

- a pawn can move one row forward (i.e. from row r to $r+1$), without changing columns;
- a rook can move any number of columns left/right without changing rows OR move any number of rows forward/backward without changing columns;
- a bishop can move to any square of the two diagonals intersecting at its currently occupied square;
- a queen can move to any square where a rook or a bishop could move to from her position;
- and a king can move to any of the 8 adjacent squares.

In the following figure, we marked by X the squares each piece can move to in a single step (here, the rows are numbered from bottom to top, and the columns from left to right).



Recently, Chess Land has become a dangerous place: pieces that are passing through the land can get captured unexpectedly by unknown forces and simply disappear. As a consequence, they would like to reach their destinations as fast (i.e. in as few moves) as possible, and they are also interested in the number of different ways it is possible for them to reach it, using

the minimal number of steps – because more paths being available could mean lower chances of getting captured. Two paths are considered different if they differ in at least one visited square.

For this problem, let us assume that pieces are entering Chess Land in a given column of row 1, and exit the land in a given column of row R. Your task is to answer Q questions: given the type of a piece, the column it enters row 1 and the column it must reach in row R in order to exit, compute the minimal number of moves it has to make in Chess Land, and the number of different ways it is able to do so.

Input

The first line contains three space separated integers R, C and Q, the number of rows and columns of Chess Land, and the number of questions, respectively. Then Q lines follow.

Each line consists of

- a character T, corresponding to the type of the piece in question ('P' for pawn, 'R' for rook, 'B' for bishop, 'Q' for queen and 'K' for king);
- two integers c_1 and c_R, $1 \leq c_1, c_R \leq C$, denoting that the piece starts from the c_1-th column of row 1, and has to reach the c_R-th column of row R.

Output

You have to print Q lines, the i-th one containing two space separated integers, the answer to the i-th question: the first one is the minimal number of steps needed, the second is the number of different paths available using this number of steps. Since the answer can be quite large, you have to compute it modulo $10^9 + 7$, using the library functions provided by the grader.

If it is impossible to reach the target square, output the line "0 0".

Library

The grader provides the following library functions for performing computations involving basic arithmetic modulo $10^9 + 7$. In all cases, the input can be any valid `int` value, and the output range is $0, 1, 2, \ldots, 10^9 + 6$. A sample implementation is available for testing your solution, see the next section for details.

- `int Add(int a, int b)`: adds numbers a and b, and returns the answer modulo 10^9+7.
- `int Sub(int a, int b)`: subtracts b from a, then returns the answer modulo $10^9 + 7$.
- `int Mul(int a, int b)`: computes the product of a and b, then returns the answer modulo $10^9 + 7$.
- `int Div(int a, int b)`: computes the quotient of a divided by $b \neq 0$ modulo $10^9 + 7$, i.e. returns the value $0 \leq q < 10^9 + 7$ if and only if `Mul(b,q)` $= a \mod (10^9 + 7)$.

You can assume that all the operations above are carried out in constant time.

To access these functions, you must add the line `#include "arithmetics.h"` to the include list of your solution.

Practice

The supplemented file *sample.zip* contains the header `arithmetics.h`, as well as the file `arithmetics.cpp`, an example implementation of the previous functions, which you should use for testing your solution.

In order to utilize these, you should copy both files to the same directory where your solution source (e.g. `chessrush.cpp`) is residing, and add the line `#include "arithmetics.h"` to its include list.

Then, simply compile `chessrush.cpp` together with `arithmetics.cpp`, e.g. using `g++ -o chessrush arithmetics.cpp chessrush.cpp` in the command line. Or, if you are using a project-based IDE, you have to manually add all three of these files to your project before building your solution.

The correct answers for the sample inputs can be found in the files `output0.txt, output1.txt`. None of the provided tools and functions check the correctness of your answers.

When submitting, you should only upload the file `chessrush.cpp` to the grading system.

Examples

Input	*Output*

`8 8 5`
`P 1 2`
`R 4 8`
`Q 2 3`
`B 3 6`
`K 5 5`
	`0 0`
	`2 2`
	`2 5`
	`2 2`
	`7 393`

Constraints

$1 \leq Q \leq 1000$
$2 \leq C \leq 1000$
$C \leq R \leq 10^9$

Time limit: 1.3 s

Memory limit: 64 MiB

Grading

Subtask	Points	Constraints
1	0	sample
2	8	$T \in \{'P', 'R', 'Q'\}$, i.e. all pieces are pawns, rooks or queens
3	15	$T = 'B'$ and $C, R \leq 100$
4	22	$T = 'B'$
5	5	$T = 'K'$ and $C, R \leq 100$ and $Q \leq 50$
6	8	$T = 'K'$ and $C, R \leq 100$
7	15	$T = 'K'$ and $C \leq 100$
8	20	$T = 'K'$
9	7	no additional constraints