Schaken

De wondere wereld van Schaakland is een grid van vierkante vakjes. Er zijn *R* rijen en *C* kolommen, en *R* is groter dan of gelijk aan *C*. De rijen en kolommen zijn genummerd van 1 tot en met *R* respectievelijk 1 tot en met *C*.

De inwoners van Schaakland worden stukken genoemd. Er zijn er vijf die het land doorkruisen: pion, toren, loper, koningin en koning. Er zijn geen paarden in Schaakland.

Elk stuk is uniek in de manier waarop het van vakje naar vakje beweegt:

- een pion beweegt een vakje vooruit (van rij *r* naar *r*+1) zonder dat de kolom verandert.
- een toren kan een willekeurig aantal vakjes naar links of naar rechts in dezelfde rij, OF, een willekeurig aantal vakjes vooruit of achteruit in dezelfde kolom.
- een loper kan naar elk vakje bewegen op de diagonalen die zijn huidige vakje doorkruisen
- een koningin kan naar elk vakje waar een toren of een loper naartoe zou kunnen vanaf het huidige vakje
- een koning kan naar elk van de 8 aangrenzende vakjes

SEE ENGLISH VERSION FOR IMAGES THAT SHOW POSSIBLE MOVEMENT FOR EACH PIECE

Recentelijk is Schaakland gevaarlijk geworden: stukken die het land doorkruisen worden soms gevangen genomen door onbekende mogendheden en verdwijnen. Dit is de reden dat de stukken graag hun bestemming zo snel mogelijk (dus: in zo min mogelijk zetten) willen bereiken. Ze zijn ook geïnteresseerd in het aantal verschillende manieren waarop ze de bestemming kunnen bereiken met dit minimale aantal zetten. Immers: meer mogelijkheden betekent een lagere kans om gevangen genomen te worden. Twee paden zijn anders als ze anders zijn in minimaal één vakje dat bezocht wordt.

Voor dit probleem geldt: de stukken komen Schaakland binnen in een kolom in rij 1, en ze verlaten Schaakland in een kolom van rij R. Jij moet Q vragen beantwoorden: gegevens het type stuk, de kolom waarin ze in rij 1 binnenkomen, en de kolom waarin ze in rij R weer vertrekken, bereken het minimale aantal zetten dat het stuk moet maken in Schaakland, en het aantal manieren waarop het stuk dit kan doen.

Invoer

Op de eerste regel van de invoer staan drie integers R, C, en Q, gescheiden door een spatie. Deze geven aan hoeveel rijen en kolommen er zijn, en hoeveel vragen je krijgt. Hierna volgen Q regels.

Op elke regel staat

- een letter T die aangeeft om welk stuk het gaat ('P' voor pion, 'R' voor toren, 'B' voor loper, 'Q' voor koningin en 'K' voor toren)
- twee integers, c_1 en c_R, $1 \leq c_1, c_R \leq C$, welke aangeven dat het stuk begin in de c_1-de kolom van rij 1, en de c_R-de kolom van rij R moet bereiken.

Uitvoer

Je moet Q regels naar de uitvoer schrijven. Op de i-de regel schrijf je twee integers, gescheiden door een spatie. Het antwoord op de i-de vraag: de eerste integer geeft aan wat het minimale aantal zetten is, de tweede hoeveel verschillende paden er zijn die dit aantal zetten hebben. Omdat het antwoord groot kan zijn, moet je het uitrekenen modulo 10^9+7. Je kunt hiervoor de library functies van de grader gebruiken.

Als het niet mogelijk is om het doelvakje te bereiken moet je "0 0" schrijven.

Library

De grader geeft je de volgende library functies om berekeningen uit te voeren modulo 10^9+7. De invoer kan elke willekeurige `int` waarde zijn, en de uitvoer valt altijd binnen het bereik [0, 1, 2, ..., 10^9+6]. Een voorbeeldimplementatie is beschikbaar om je oplossing te testen.

- `int Add(int a, int b)`: somt de getallen *a* en *b* en levert het antwoord modulo 10^9+7.
- `int Sub(int a, int b)`: trekt *b* van *a* af, en levert het antwoord modulo 10^9+7.
- `int Mul(int a, int b)`: berekent het product van *a* en *b*, en levert het antwoord modulo 10^9+7.
- `int Div(int a, int b)`: deelt *a* door *b* (*b* != 0) en levert het resultaat modulo 10^9+7. Oftewel: levert als resultaat de waarde $0 \le q < 10^9+7$, dan en slechts dan als het resultaat van `Mul(b,q)` gelijk is aan *a* modulo 10^9+7.

Je kunt ervan uitgaan dat de bovenstaande operaties in constante tijd worden uitgevoerd.

Om deze functies te kunnen gebruiken moet je de regel `#include "arithmetics.h"` toevoegen aan de includes van je oplossing.

Oefenen

Het bestand *sample.zip* bevat de header `arithmetics.h` en ook het bestand `arithmetics.cpp`, een voorbeeldimplementatie van de genoemde functies die je kunt gebruiken om je oplossing te testen.

Om deze bestanden te gebruiken moet je ze beide kopiëren naar de directory waar je broncode (dus `chessrush.cpp`) staat, en dan de regel `#include "arithmetics.h"` toevoegen aan de lijst met includes.

Daarna compileer je `chessrush.cpp` gezamenlijk met `arithmetics.cpp`, bijvoorbeeld middels `g++ -o chessrush arithmetics.cpp chessrush.cpp`. Of, als je een project-

based IDE gebruikt moet je al deze drie files aan je project toevoegen voordat je kunt builden.

De correcte antwoorden voor de voorbeeld invoer staat in de bestanden `output0.txt`, `output1.txt`. Geen van de aangeleverde tools of functies controleert of je programma correct is.

Als je je programma inzendt moet je alleen het bestand `chessrush.cpp` uploaden naar het grading systeem.

Voorbeelden, randvoorwaarden, etc.

Zie Engelse tekst.

Chess Rush

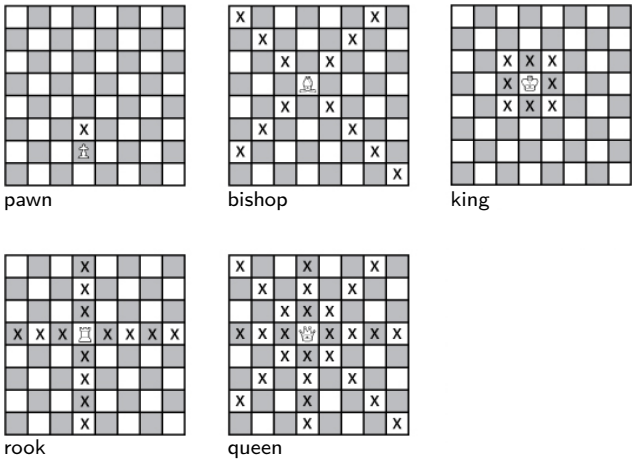
The mythic world of Chess Land is a rectangular grid of squares with R rows and C columns, R being greater than or equal to C. Its rows and columns are numbered from 1 to R and 1 to C, respectively.

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