HPC^3 2024
Problem I, English
King of the Hunt
Maximum Points: 70

Your father, who was a famed hunter of chess pieces, has recently passed away without completing his lifelong mission of clearing the continent of all wild chess pieces. You wish to complete his mission with nothing but your fathers map, his “piecetiary”, and his kit of piece-catching devices.
You will be hunting pieces on an $N \times M$ grid $(1 \leq N \leq 10^4,\ 1 \leq M \leq 10^4)$. A piece exists at a single point on the grid.
The hunt will progress like this:

- The first piece will be placed at a random point on the grid
- Until you catch it, hunt rounds will occur: You will place a device at a point on the grid, then the piece will move.
- Once you catch it, all devices will be removed, and the next piece will be placed on the grid at random.

There are 5 types of pieces that each move in a distinct pattern along the grid:

- Pawns move to cardinally adjacent points.
- Kings move to cardinally or diagonally adjacent points.
- Bishops move any number of points diagonally in a single direction.
- Rooks move orthogonally to any point in their row or column.
- Queens move orthogonally to any point in their row or column or any number of points diagonally in a single direction.

You have two types of devices available to you: Traps and sensors. Traps catch any pieces that move through or on them, but pieces will never do so unless they must. Sensors alert you whenever a piece moves through or on them or a square cardinally or diagonally adjacent to them.
Aside from avoiding traps, pieces are simple creatures, they must choose a random point each hunt round from their available points and move to it.
You have limited food and so must catch all pieces in 500,000 hunt rounds.

Notes

- This problem is interactive, for every test case your program will be repeatedly giving output and receiving input that is dependent on past outputs.
- This problem has randomness involved. Because of this, it may manifest nondeterministic results. However, the correct solution will always be the most optimal and so will always score the maximum number of points.

Subproblem 1

Your fathers “piecetiary” contains an ordered list of all the pieces you will be hunting represented by a string S of length p $(1 \leq p \leq 50)$ where each character c in the string represents a piece and when it will arrive (“P” for pawn, “K” for king, “B” for bishop, “R” for rook, and “Q” for queen).

Given N, M, and S. Preform hunt rounds to capture all pieces within 500,000 hunt rounds.

First input format

The first line of each input contains 3 integers N, M and p.
The second line of each input contains a string of length p: S.

```
N  M  p
S
```

Hunt round output format

The first line of each output contains 1 binary value D.
The second line of each output contains 2 integers: x and y.

```
D
x  y
```

D is the device you will place (0 for trap, 1 for sensor) and x, y is the position of the device. If there is already a device there, it will be replaced.

Hunt round input format

The first line of each input contains 2 integers R and k.
The second line of each input contains k integer pairs: The content of array P.

```
R  k
P[0][0]  P[0][1]  P[1][0]  P[1][1]  …  P[k-1][0]  P[k-1][1]
```

R is the result of the pieces movement, between -1 and 3.

- If -1, you have exceeded to 500,000 round limit and are out of time.
- If 0, the piece moved and nothing else happened.
- If 1, the piece triggered some sensors. This means that every pair in P is a point within the range of a sensor that a piece moved through or to.
- If 2, the piece was captured. Either it was forced to step into a trap, or the last device placed was a trap at its location. All devices are removed.
- If 3, the piece was captured and was the last piece, meaning you win!

Example Test Cases

Input 1

```
3  3  2
PK
```

Output H1 : R1

```
1
2  2
```

Input H1 : R1

```
1  1
1  2
```

Output H1 : R2

```
0
1  2
```

Input H1 : R2

```
2  0
```

Output H2 : R1

```
1
2 2
```

Input H2 : R1

```
1 2
2 1 3 1
```

Output H2 : R2

```
0
3 1
```

Input H2 : R2

```
3 0
```

You are hunting a pawn, then a rook, on a 3x3 grid. The optimal move in all cases is to place a sensor in the middle of the grid, because this will tell you the exact location of the piece after it moves, allowing you to trap it.

Subproblem 2

After clearing all of the pieces in your home, you have moved on to other lands. The pieces here have evolved electro-telekinetic powers, this leads to 3 differences to the pieces in subproblem 1: First, they are too dangerous to approach, so you cannot place traps at their current location. Second, they can remotely deactivate sensors, so you will not be placing any. Finally, they are surrounded by a bright halo, meaning you know their location at all times.

Your fathers "piecetiary" works the same as in subproblem 1. The maximum value of N is 100, and the maximum value of M is 100.

Given N, M, and S. Preform hunt rounds to capture all pieces within 5,000,000 hunt rounds.

First input format

The first line of each input contains 5 integers N, M, p, x, and y.
The second line of each input contains a string of length p: S.

```
N  M  p  x  y
S
```

Where x, y is the starting point of the piece.

Hunt round output format

The first and only line of each output contains 2 integers: x and y.

```
x  y
```

x, y is the position of the trap you will place.

Hunt round input format

The first line of each input contains 1 integer R.
The second line of each input contains 2 integers: x and y.

```
R
x  y
```

R is the result of the pieces movement, between -1 and 2.

- If -1, you have exceeded to 5,000,000 round limit and are out of time.
- If 0, the piece moved to x, y.
- If 1, the piece was captured. All traps are removed and x, y is the starting position of the next piece.
- If 2, the piece was captured and was the last piece, meaning you win!

Example Test Cases

Input 1

```
3  3  2  1  1
PK
```

Output H1 : R1

```
2  2
```

Input H1 : R1

```
0
2  1
```

Output H1 : R2

```
1  1
```

Input H1 : R2

```
0
3  1
```

Output H1 : R3

```
3  2
```

Input H1 : R3

```
0
2  1
```

Output H1 : R4

```
3  1
```

Input H1 : R4

```
1
2  2
```

Output H2 : R1

```
1  2
```

Input H2 : R1

```
0
1  1
```

Output H2 : R2

```
2  2
```

Input H2 : R2

```
0
2  1
```

Output H2 : R3

```
3  2
```

Input H2 : R3

```
0
3  1
```

Output H2 : R3

```
2  1
```

Input H2 : R3

```
2
99999999999999999999999999999999999  999999999999999999999999999999999999
```

Note that the king is caught faster despite being a more mobile piece because it randomly made moves that caused it to enter a worse position.