PacMan - BFS

In the previous game, you performed DFS on the PacMan grid. It was seen that DFS might not always give the shortest path between the source and the destination.

In this game, PacMan is positioned in a grid. PacMan has to find the food using Breadth First Search (BFS), provided the grid is completely observable, perform a BFS on the grid and then print the path obtained by BFS from the PacMan to the food.

Input Format

The first line contains 2 space separated integers which is the position of the PacMan.
The second line contains 2 space separated integers which is the position of the food.
The third line of the input contains 2 space separated integers. Indicating the size of the rows and columns respectively.
This is followed by *row* (r) lines each containing *column* (c) characters. A wall is represented by the character '%' (ascii value 37), PacMan is represented by UpperCase alphabet 'P' (ascii value 80), empty spaces which can be used by PacMan for movement is represented by the character '-' (ascii value 45) and food is represented by the character '.' (ascii value 46)

The top left of the grid is indexed (0,0) and the bottom right of the grid is indexed (r-1,c-1)

The grid is indexed as per matrix convention

Populating Queue

In order to maintain uniformity across submissions, please follow the below mentioned order in pushing nodes to queue. If a node has all the 4 adjacent neighbors. Then,

1. UP
2. LEFT
3. RIGHT
4. DOWN

UP goes first into the queue, followed by **LEFT**, followed by **RIGHT** and then by **DOWN**.

so, if (1,1) has all its neighbors not visited, (0,1), (1,0), (1,2), (2,1) then,

1. (0,1) - UP
2. (1,0) - LEFT
3. (1,2) - RIGHT
4. (2,1) - DOWN

So, (0,1) is the first to be popped from the queue.

Output Format

Each cell in the grid is represented by its position in the grid (r,c). PacMan can move only UP, DOWN, LEFT or RIGHT. Your task is to print all the nodes that you encounter while printing BFS tree. While populating the queue, the following convention must be followed.

```
 %
%--
 -
```

In the above cell, LEFT and UP are invalid moves. You can either go RIGHT or DOWN. RIGHT is populated first followed by DOWN. i.e., populate the queue UP, LEFT, RIGHT and DOWN order so that UP gets popped first from the queue.

Print N, saying N nodes are expanded using BFS. In the next line, Starting from the source node 'P' (including it), print all the nodes (r,c) expanded using BFS each node in a new line (r,c) until the food node is expanded.

```
N
r c
r1 c1
....
```

Then, print the distance 'D' between the source 'P' and the destination '.' calculated using BFS. D+1 lines follow, each line having a node encountered between 'P' and '.' both included. D+1 lines essentially representing the path between source and the destination.

Sample Input

```
3 9
5 1
7 20
%%%%%%%%%%%%%%%%%%%%
%--------------%---%
%-%%-%%-%%-%%-%%-%-%
%--------P-------%-%
%%%%%%%%%%%%%%%%%%-%
%.-----------------%
%%%%%%%%%%%%%%%%%%%%
```

Sample Output

sample output

In this example, PacMan is at the position (3,9) and the food is at the position (5,1). BFS tree is printed starting from (3,9) until the food node is expanded. The BFS path length between (3,9) and (5,1) is 32. All the nodes encountered between (3,9) and (5,1) both included is printed in the next 33 lines.

Task

Your task is to complete the function nextMove that takes in r,c as the grid size, pacman_r and pacman_c which is the position of Pacman and food_r and food_c as the position of food and the grid as input and print the output in the required format.