Pacman - UCS

In the previous game, you performed BFS on the PacMan grid. In this game we use uniform cost search or Cheapest First search to find an efficient path between pacman and 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

For the sake of uniformity across all codes, cost to reach a neighboring node

- 0 if a food is present.
- 1 otherwise.

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 the path between source and destination calculated using UCS.

Print the distance 'D' between the source 'P' and the destination '.' calculated using UCS. 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

```
35 35
35 1
37 37
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%-------%-%-%----------%---%-----%-%
%-%%%%%%%-%-%%%-%-%%%-%%%-%%%%%%%-%-%
%-------%-------%-%-----%-----%-%---%
%%%%%-%%%%%-%%%-%-%-%%%-%%%%%-%-%%%
%---%-%-%-%---%-%-%-%---%-%---%-%---%
%-%%%-%-%-%%%-%%%%%-%%%-%-%%%-%%%-%
%-------%-----%---%---%-----%-%-%---%
%%%-%%%%%%%%%-%%%%%%%-%%%-%%%-%-%-%-%
%------------%-------%-%---%-----%-%
%-%-%%%%%-%-%%%-%-%-%%%-%-%%%-%%%-%-%
%-%-%-----%-%-%-%-%-----%---%-%-%-%-%
%-%-%-%%%%%%%-%-%%%%%%%%%%-%%%-%-%%%-%
%-%-%-%-----%---%-----%-----%---%---%
%%%-%%%-%-%%%%%-%%%%%-%%%-%%%-%%%%%-%
%-----%-%-%-----%-%-----%-%---%-%-%-%
%-%-%-%-%%%-%%%-%%%-%%%-%-%-%-%-%-%
%-%-%-%-%-----------------%-%-%-----%
%%%-%%%%%%%-%-%-%%%%%-%%%-%-%%%-%%%%%
%-------%-%-%-%-----%---%-----%-%---%
%%%%%-%-%-%%%%%%%%%%-%%%%%%%%%%%%%-%-%%%
%---%-%----------%-%-----%---%-%---%
%-%%%-%%%%%-%%%%%%%%%%-%%%%%-%-%-%%%-%
%-%---%------%--------%-----%-------%
%-%-%-%%%%%-%%%-%-%-%-%%%%%%%%%%%%%%%
%-%-%---%-----%-%-%-%-------%---%-%-%
%-%-%%%-%%%-%-%-%-%%%%%%%%%%-%%%-%-%-%
```

```
%-%---%-%---%-%-%---%-%---%-%-%-----%
%-%%%-%%%-%%%%%-%%%-%-%-%%%%%-%-%%%%%
%-------%---%-----%-%-----%---%-%---%
%%%-%-%%%%%-%%%%%-%%%-%%%-%-%%%-%-%%%
%-%-%-%-%-%-%-%-----%-%---%-%---%-%-%
%-%-%%%-%-%-%-%-%%%%%%%%%-%-%-%-%-%-%
%---%---%---%----------%-----%
%-%-%-%-%%%-%%%-%%%%%%%%-%%%-%%%-%%%-%
%.%-%-%-------%---%-------%---%-%--P%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Sample Output

sample output

In this example, PacMan is at the position (35,35) and the food is at the position (35,1). UCS tree is expanded starting from (35,35) until the food node is expanded. The UCS path length between (35,35) and (35,1) is 210. All the nodes encountered between (35,35) and (35,1) both included is printed in the next 211 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.