Experiment No. 9

Semester	T.E. Semester VI
Subject	ARTIFICIAL INTELLIGENCE (CSL 604)
Subject Professor In-charge	Prof. Avinash Shrivas
Assisting Teachers	Prof. Avinash Shrivas

Student Name	Deep Salunkhe
Roll Number	21102A0014
Lab Number	310A

Title:

Maze Escape

Theory:

Maze Escape is a popular 1 player game where a bot is trapped in a maze and is expected to find its way out. In this version of the game, 2 bots whose positions are not visible to each other are placed at random points in the maze. the first bot to find its way out of the maze wins the game.
The visibility of the bot is restricted to its 8 adjacent cells as shown in the figure below

```
---
-b-
---
```

where *b* is the bot. Bots can be facing any of the 4 directions. To make this game more interesting, the orientation of both the bots are randomly chosen at the start of the game and the map visible to them also changes according to the orientation.
If the actual map is as shown below,

```
#######
#--#-b#
#--#--#
#--#--#
e-----#
#-----#
#######
```

and your bot is positioned at (1,5) and is facing the RIGHT side of the maze, the input will be

###
#b-
#--
If your bot is facing UP, your input will be
###
-b#
--#
If your bot is facing LEFT, your input will be
--#
-b#
###
And if your bot is facing DOWN, your input will be
#--
#b-
###
It is to be noted that your bot faces the direction in which it chooses to make its next move.

Input Format

The walls are represented by character # (ascii value 35), empty cells are represented by - (ascii value 45), the maze door which is the indication of the way out is represented by *e* (ascii value 101)
The input contains 4 lines, the first line being the player id (1 or 2) and followed by 3 lines, each line containing 3 of the above mentioned characters which represents the 8 adjacent cells of the bot. The visible maze given as input always has the bot at the center.

Constraints

5 <= r,c <= 30 where r, c is the number of rows and columns of the maze.

Output Format

Output *UP*, *DOWN*, *LEFT* or *RIGHT* which is the next move of the bot.

Sample Input

1
###
#--
#--

Sample Output

RIGHT

Explanation

Considering the maze given above. If the input is as given below with the bot initially facing the RIGHT side of the maze, if the bot chooses to go RIGHT, the new position of the bot in the maze would be
#######
#--#--#
#--#-b#
#--#--#
e-----#
#-----#
#######
The bot moves 1 step DOWN w.r.t the maze. As the bot is facing DOWN side of the maze, his next input would be
#--
#--
#--
with the bot at the center.

Implementation:

```
def writeMap(view, name):
    f = open(name,'w')
    for i in range(0,len(view)): view[i] = view[i]+'\n'
    f.writelines(view)
    f.close()

def writePrev(output):
    f = open('prev.dat','a+')
    f.write(output+'\n')
    f.close()
    return output


def nextMove(view):
    target = []
    for i in range(0, len(view)):
        if 'e' in view[i]:
            target = [view[i].index('e'),i]
            if target[0] == 0: return "LEFT"
            elif target[0] == 2: return "RIGHT"
            elif target[0] == 1: return "UP"
        if '-' in view[i]:
            for j in range(view[i].index('-'),3):
                if i == 1 and j == 1: j += 1
                if '-' in view[i][j:]: target.append([view[i].index('-',j),i])

    f = open('prev.dat','a+')
    f.close()
    f = open('prev.dat','r')
    pMoves = f.readlines()
    f.close()
    f = open('prevMap.dat','a+')
    f.close()
    f = open('prevMap.dat','r')
    pMap = f.readlines()
    f.close()
    f = open('prevMap2.dat','a+')
    f.close()
    f = open('prevMap2.dat','r')
    pMap2 = f.readlines()
    for i in range(0,len(pMap)): pMap[i] = pMap[i].replace('\n','')
    for i in range(0,len(pMap2)): pMap2[i] = pMap2[i].replace('\n','')


    #remove duplicates
```

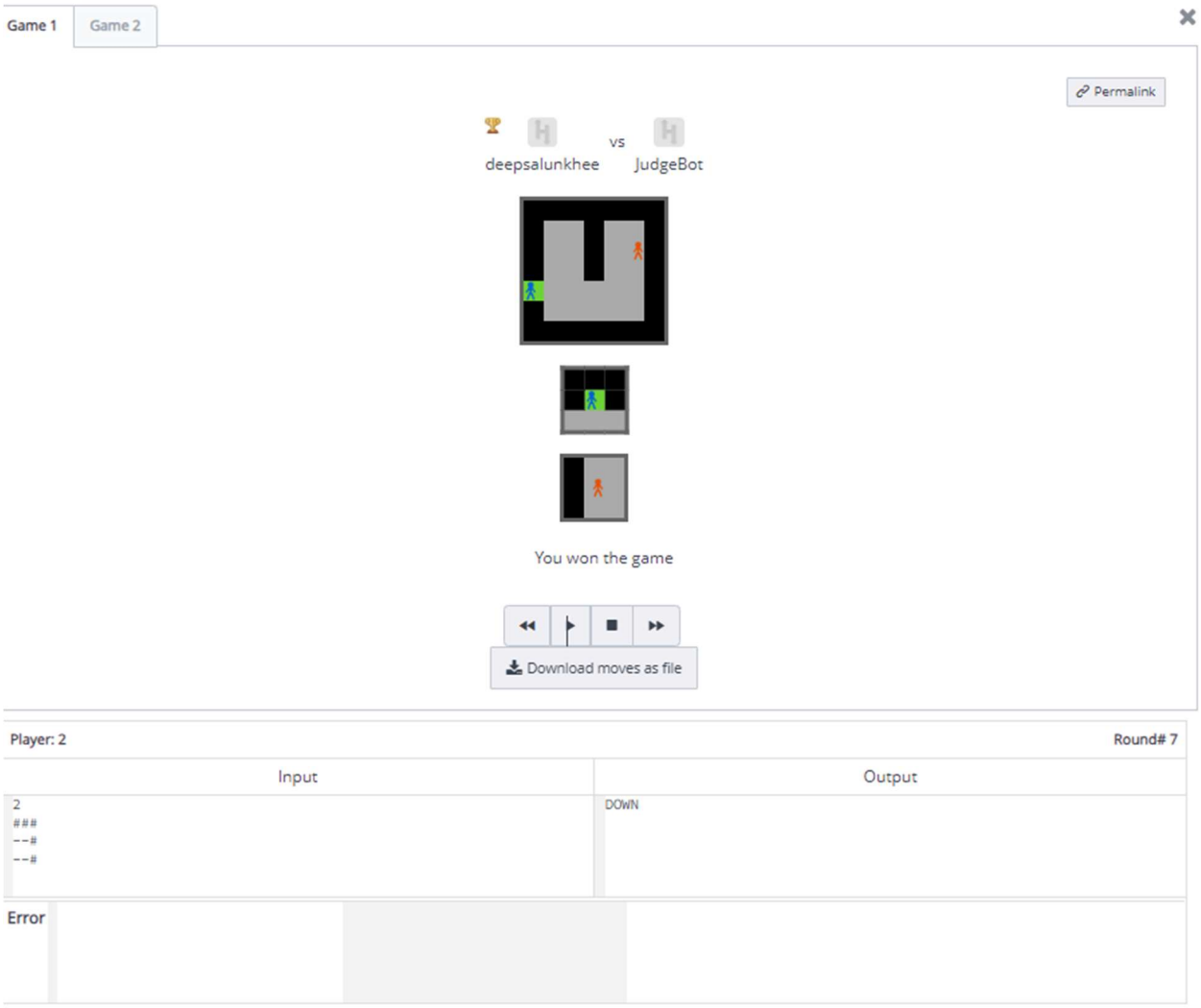
```
    dup = target
    target = []
    [target.append(x) for x in dup if x not in target]


    if len(pMoves) < 3 and len(target) == 3:
        if [0,1] in target and [1,2] in target:
            writeMap(view, 'prevMap.dat')
            writeMap(pMap,'prevMap2.dat')
            return writePrev("LEFT")
        if [1,2] in target and [2,1] in target:
            writeMap(view, 'prevMap.dat')
            writeMap(pMap,'prevMap2.dat')
            return writePrev("DOWN")
        if [2,1] in target and [1,0] in target:
            writeMap(view, 'prevMap.dat')
            writeMap(pMap,'prevMap2.dat')
            return writePrev("RIGHT")
        if [1,0] in target and [0,1] in target:
            writeMap(view, 'prevMap.dat')
            writeMap(pMap,'prevMap2.dat')
            return writePrev("UP")
    if view == ['---','---','--#'] and pMap == ['---','--#','--#']:
        writeMap(view, 'prevMap.dat')
        writeMap(pMap,'prevMap2.dat')
        return writePrev("RIGHT")
    if view == ['#--','#--','#--'] and pMap == ['#--','#--','#--'] and pMap2 ==
['#--','#--','#--']:
        writeMap(view, 'prevMap.dat')
        writeMap(pMap,'prevMap2.dat')
        return writePrev("RIGHT")
    if [1,0] in target:
        writeMap(view, 'prevMap.dat')
        writeMap(pMap,'prevMap2.dat')
        return writePrev("UP")
    if [2,1] in target:
        writeMap(view, 'prevMap.dat')
        writeMap(pMap,'prevMap2.dat')
        return writePrev("RIGHT")
    if [1,2] in target:
        writeMap(view, 'prevMap.dat')
        writeMap(pMap,'prevMap2.dat')
        return writePrev("DOWN")
    if [0,1] in target:
        writeMap(view, 'prevMap.dat')
        writeMap(pMap,'prevMap2.dat')
        return writePrev("LEFT")
```

```
id = int(input())
view = []
for i in range(0,3):
    view.append(input())

print(nextMove(view))
```

Output:



Maze Escape

Problem | Submissions | Leaderboard | Discussions

Submitted a day ago

Challenge another hacker

Download All Games

Opponent	Results (Won/Tied/Lost)	Games
karnouk	2/0/0	View Game
rafasj6	2/0/0	View Game
johnstmathai	2/0/0	View Game