```
# -*- coding: utf-8 -*-
"""
Created on Fri Jul  3 14:43:05 2020

@author: Parshva Timbadia
"""
import random

"""
This is extaction Module
"""

Data=[]

def is_number(s):
    try:
        float(s)
        return True
    except ValueError:
        return False


with open('environ.txt', 'r') as myfile:

    for line in myfile:

        for word in line.split():

            if is_number(word):

                Data.append(word)


'Extracting Value for the Grid '

GRID=[]
for i in range(2):
    GRID.append(int(Data.pop(0)))



'Extracting Value for the Initial'

INITIAL =[]
for i in range(2):
    INITIAL.insert(0,int(Data.pop(-1)))

#Assigning Values to the INITIAL


INITIAL_y= INITIAL[0] -1
INITIAL_x= INITIAL[1] -1



'Extracting the value for the Moves'

MOVES= int(Data.pop(-1))



'Now converting Data into the Matrix Format as a part of the Gird'
DIRT=[]
for i in range(GRID[0]):
    NEW_LIST =[]
    for j in range(GRID[1]):


        NEW_LIST.append(float(Data.pop(0)))

    DIRT.append(NEW_LIST)

"""

Extraction Module Ends here
Variables Declared:

GIRD:
DIRT:
MOVIES:
INITIAl:

"""

#Now defining a class vaccum that would help moving around and cleaning
COLLECT=0 #Indicates the amount of DIRT Collected
```

```
def UP():
    global INITIAL_y, COLLECT
    if INITIAL_y > 0:
        INITIAL_y = INITIAL_y -1

    print("U:  ", COLLECT)

def DOWN():
    global INITIAL_y,  COLLECT
    if INITIAL_y < len(DIRT)-1:
        INITIAL_y = INITIAL_y +1

    print("D:  ", COLLECT)


def LEFT():
    global INITIAL_x,  COLLECT
    if INITIAL_x > 0:
        INITIAL_x = INITIAL_x - 1
    print("L:  ", COLLECT)

def RIGHT():
    global INITIAL_x, COLLECT
    if INITIAL_x < len(DIRT[0])-1:
        INITIAL_x = INITIAL_x + 1
    print("R:  ", COLLECT)


def CLEAN():    #Cleans the tile and makes the value at that point as ZERO
    global INITIAL_y, INITIAL_x,DIRT, COLLECT


    COLLECT+= DIRT[INITIAL_y][INITIAL_x]
    DIRT[INITIAL_y][INITIAL_x]=0

    print('S:  ', COLLECT)



def Random(DIRT, MOVES, COLLECT):
    global INITIAL_y, INITIAL_x
    initial_position =0
    Movement=[0,1,2,3]


    #We will loop through until the number of moves are satisfied
    while initial_position < MOVES:

        #We will randomly select the Movement


        i=random.choice(Movement)
        if INITIAL_y>0 and (i==0):

            CLEAN()
            initial_position+=1

            if initial_position%5==0:
                display(DIRT, INITIAL_y, INITIAL_x)



            UP()
            initial_position+=1

            if initial_position%5==0:
                display(DIRT, INITIAL_y, INITIAL_x)




        elif INITIAL_y< len(DIRT)-1 and(i==1):
            CLEAN()
            initial_position+=1

            if initial_position%5==0:
                display(DIRT, INITIAL_y, INITIAL_x)

            DOWN()
            initial_position+=1

            if initial_position%5==0:
                display(DIRT, INITIAL_y, INITIAL_x)
        elif INITIAL_x > 0 and (i==2):
            CLEAN()
            initial_position+=1

            if initial_position%5==0:
                display(DIRT, INITIAL_y, INITIAL_x)


            LEFT()
            initial_position+=1

            if initial_position%5==0:
                display(DIRT, INITIAL_y, INITIAL_x)






        else:
            CLEAN()
            initial_position+=1

            if initial_position%5==0:
                display(DIRT, INITIAL_y, INITIAL_x)


            RIGHT()
            initial_position+=1

            if initial_position%5==0:
                display(DIRT, INITIAL_y, INITIAL_x)








def Greedy( DIRT, MOVES, COLLECT):
    initial_position=0
    global INITIAL_y, INITIAL_x


    while initial_position < MOVES:

        #We will randomly select the Movement
        CLEAN()
        initial_position+=1

        if initial_position%5==0:
                display(DIRT, INITIAL_y, INITIAL_x)


        result_of_successor = successor(INITIAL_y, INITIAL_x, DIRT)


        '''
            The below for loop with provide us with the list something
            like  [1,2,0]
        '''
        result_of_first_val = []
        for j in range(len(result_of_successor)):
                result_of_first_val.append(result_of_successor[j][0])


        if check(result_of_first_val):
            '''
            This IF statement will only run if the vaccum get occupied by the same
            value and this will help it to JUMP in ramdom direction due to the shuffle
            function.
            '''


            '''
            The below for loop with provide us with the list something
            like  ['up','down', 'left']
            '''
            options=[]
            for j in range(len(result_of_successor)):
                options.append(result_of_successor[j][1])

            #Now shuffling the options list and selecting the first value
            random.shuffle(options)
            value = options[0]
            #Now select the directions
            if value == 'up':
                UP()
            elif value == 'down':
                DOWN()
            elif value =='left':
                LEFT()
            else:
                RIGHT()


            initial_position+=1

            if initial_position%5==0:
                display(DIRT, INITIAL_y, INITIAL_x)


        else:
            '''
            This will be only executed when we are only surrounded by different values like
            [1,2,4,6] and it will select the max value and make a JUMP in that direction.

            '''
            optimal_val= max(result_of_successor)

            # print(successor(INITIAL_y, INITIAL_x , DIRT))
            # print('Value:', optimal_val)

            #This provies us the shuffled results incase the values are same


            #Now Jump for the Direction:

            if INITIAL_y>0 and optimal_val[0]==DIRT[INITIAL_y-1][INITIAL_x]:
                UP()
            elif INITIAL_y< len(DIRT)-1 and optimal_val[0]== DIRT[INITIAL_y+1][INITIAL_x]:
                DOWN()
            elif  INITIAL_x > 0 and optimal_val[0] == DIRT[INITIAL_y][INITIAL_x-1]:
                LEFT()
            else:
                RIGHT()


        initial_position+=1

        if initial_position%5==0:
                display(DIRT, INITIAL_y, INITIAL_x)




def successor(INITIAL_y, INITIAL_x, DIRT):

        #Dont Change Anything Here
        '''
        Function that adds the elements/Dirt grid around in the successor list.

        '''

        successor=[]

        if INITIAL_y > 0:
           successor.append((DIRT[INITIAL_y -1][INITIAL_x], "up")) #Going UP

        if INITIAL_y < len(DIRT)-1:
           successor.append((DIRT[INITIAL_y+1][INITIAL_x], "down"))    #Going Down

        if INITIAL_x > 0:
           successor.append((DIRT[INITIAL_y][INITIAL_x-1],"left")) #Going Left

        if INITIAL_x < len(DIRT[0])-1:
           successor.append((DIRT[INITIAL_y][INITIAL_x +1],"right")) #Going Right


        return successor


def check(list):
    '''
    Function to check all the values in the list are same or not.

    '''

    return all(i == list[0] for i in list)
```

```
def optimal(DIRT, MOVES, COLLECT):
    initial_position=0
    global INITIAL_y, INITIAL_x

    visited={}


    while initial_position < MOVES:

        result_of_successor = successor(INITIAL_y, INITIAL_x, DIRT)



        optimal_sol= max(result_of_successor)


        result_of_first_val = []
        for j in range(len(result_of_successor)):
                result_of_first_val.append(result_of_successor[j][0])


        if check(result_of_first_val):
            '''
            This IF statement will only run if he vaccum get occupied by the same
            value and this will help it to JUMP in ramdom direction due to the shuffle
            function.
            '''


            '''
            The below for loop with provide us with the list something
            like  ['up','down', 'left']
            '''
            options=[]
            for j in range(len(result_of_successor)):
                options.append(result_of_successor[j][1])

            #Now shuffling the options list and selecting the first value
            random.shuffle(options)
            value = options[0]

            #Now select the directions
            if value == 'up':
                UP()
            elif value == 'down':
                DOWN()
            elif value =='left':
                LEFT()
            else:
                RIGHT()


            initial_position+=1

            if initial_position%5==0:
                display(DIRT, INITIAL_y, INITIAL_x)

            CLEAN()
            initial_position+=1

            if initial_position%5==0:
                display(DIRT, INITIAL_y, INITIAL_x)


        else:

            if INITIAL_y>0 and optimal_sol[0]==DIRT[INITIAL_y-1][INITIAL_x] and (INITIAL_y-1, INITIAL_x) not in visited:
                UP()
                initial_position +=1

                if initial_position%5==0:
                    display(DIRT, INITIAL_y, INITIAL_x)


                visited[(INITIAL_y, INITIAL_x)]= DIRT[INITIAL_y][INITIAL_x]

                if  visited[(INITIAL_y, INITIAL_x)] !=0:
                    CLEAN()
                    initial_position+=1

                    if initial_position%5==0:
                        display(DIRT, INITIAL_y, INITIAL_x)
            elif INITIAL_y< len(DIRT)-1 and optimal_sol[0]== DIRT[INITIAL_y+1][INITIAL_x] and (INITIAL_y+1, INITIAL_x) not in visited:
                DOWN()
                initial_position +=1

                if initial_position%5==0:
                    display(DIRT, INITIAL_y, INITIAL_x)

                visited[(INITIAL_y, INITIAL_x)]= DIRT[INITIAL_y][INITIAL_x]

                if  visited[(INITIAL_y, INITIAL_x)] !=0:
                    CLEAN()
                    initial_position+=1

                    if initial_position%5==0:
                        display(DIRT, INITIAL_y, INITIAL_x)



            elif  INITIAL_x > 0 and optimal_sol[0] == DIRT[INITIAL_y][INITIAL_x-1] and (INITIAL_y, INITIAL_x-1) not in visited:
                LEFT()
                initial_position +=1

                if initial_position%5==0:
                        display(DIRT, INITIAL_y, INITIAL_x)

                visited[(INITIAL_y, INITIAL_x)]= DIRT[INITIAL_y][INITIAL_x]

                if  visited[(INITIAL_y, INITIAL_x)] !=0:
                    CLEAN()
                    initial_position+=1

                    if initial_position%5==0:
                        display(DIRT, INITIAL_y, INITIAL_x)


            elif  (INITIAL_y, INITIAL_x+1) not in visited:
                RIGHT()
                initial_position +=1

                if initial_position%5==0:
                        display(DIRT, INITIAL_y, INITIAL_x)

                visited[(INITIAL_y, INITIAL_x)]= DIRT[INITIAL_y][INITIAL_x]

                if  visited[(INITIAL_y, INITIAL_x)] !=0:
                    CLEAN()
                    initial_position+=1

                    if initial_position%5==0:
                        display(DIRT, INITIAL_y, INITIAL_x)

            else:
                '''
                This means all the neighbous has been marked as visited and it should jump in random
                direction.

                    '''
                options=[]
                for j in range(len(result_of_successor)):
                    options.append(result_of_successor[j][1])

                #Now shuffling the options list and selecting the first value
                random.shuffle(options)
                value = options[0]

                #Now select the directions
                if value == 'up':
                    UP()
                elif value == 'down':
                    DOWN()
                elif value =='left':
                    LEFT()
                else:
                    RIGHT()


                initial_position+=1

                if initial_position%5==0:
                        display(DIRT, INITIAL_y, INITIAL_x)



def display(DIRT, INITIAL_y, INITIAL_x):
    '''
        This function will be printed only when the steps are in multiple of 5.
    '''



    Matrix = DIRT

    for i in range(len(Matrix)):
        for j in range(len(Matrix[0])):

            if i==INITIAL_y and j== INITIAL_x:
                print("[", float(Matrix[i][j]),']', end='  ')
            else:
                print(float(Matrix[i][j]), end='  ')

        print()



'''

TO RUN THE TASK 1: UNCOMMENT THE CODE BELOW

'''
# Random(DIRT, MOVES, COLLECT)
# print("Total Amount of Dirt Collected:", COLLECT)
'''

TO RUN THE TASK 2: UNCOMMENT THE CODE BELOW

'''
# Greedy(DIRT, MOVES, COLLECT)
# print("Total Amount of Dirt Collected:", COLLECT)
'''

TO RUN THE TASK 3: UNCOMMENT THE CODE BELOW

'''
# optimal(DIRT, MOVES, COLLECT)
# print("Total Amount of Dirt Collected:", COLLECT)
```