You can view this report online at : https://www.hackerrank.com/x/tests/1465744/candidates/52265660/report

| | |
|---|---|
| Email: | hoonjung06@gmail.com |
| Test Name: | **ACSL 2022-23, Contest 4, Intermediate Division Programming Problem** |
| Taken On: | 9 May 2023 16:10:17 PDT |
| Time Taken: | 1813 min 53 sec/ 4320 min |
| Invited by: | ACSL Contests |
| Invited on: | 9 May 2023 14:53:04 PDT |
| Skills Score: | |
| Tags Score: | |

**100%**

**5/5**

scored in **ACSL 2022-23, Contest 4, Intermediate Division Programming Problem** in 1813 min 53 sec on 9 May 2023 16:10:17 PDT

**Recruiter/Team Comments:**

*No Comments.*

| | Question Description | Time Taken | Score | Status |
|---|---|---|---|---|
| Q1 | Targets > Coding | 2 hour 7 min 20 sec | 5/ 5 | ✓ |

**QUESTION 1**

Correct Answer

Score 5

## Targets > Coding

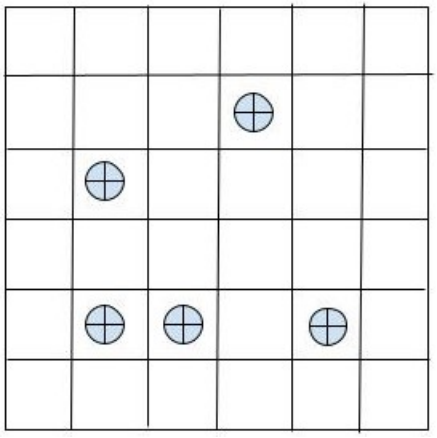
QUESTION DESCRIPTION

### PROBLEM STATEMENT:

Given the *n* x *n* grid with target locations and with empty border cells, place arrows in every border cell and find the target(s) that get(s) hit the most. The arrow stops when it hits a target. The arrows that can be placed in any border cell are defined as A-H: (A) ←, (B) ↑, (C) →, (D) ↓, (E) ↖, (F) ↗, (G) ↘, and (H) ↙. Output the location(s) of the target(s) as 2-character strings separated by a single space in row-column order. The upper left corner is identified as location (0,0) using row-major order.

### EXAMPLE:

In the grid below, the size is 6 x 6 with 5 different targets.



By checking every border position, the following arrows will hit each target:

| Target | Border cell and direction | Border cell and direction | Border cell and direction | Border cell and direction | Border cell and direction | Border cell and direction | Border cell and direction | Border cell and direction |
|---|---|---|---|---|---|---|---|---|
| (1,3) | 02G | 03D | 04H | 10C | 15A | 35E | 40F | 53B |
| (2,1) | 01D | 03H | 10G | 20C | 25A | 30F | 54E | |
| (4,1) | 05H | 30G | 40C | 50F | 51B | 52E | | |
| (4,2) | 02D | 15H | 20G | 51F | 52B | 53E | | |
| (4,4) | 00G | 04D | 35H | 45A | 53F | 54B | 55E | |

The output is “13” since that target can be hit by a maximum of 8 different arrows.

## TASK:

Complete the function **targetsWithMostArrows** that is called from a program that inputs the following data as its parameters and outputs the following information for each individual input:

- The function has 2 parameters:  a number, *size*, representing the size of the grid and a string, *targets,* representing the locations of the targets as 2-character strings separated by a single space
- The function should return a string of 2-character strings, in row-column order. representing the location of the target(s) that could be hit by the most arrows without encountering another target

You may create additional functions that are called from **targetsWithMostArrows** if needed in solving the problem.

## CONSTRAINTS:

The inputted number will be no more than 10.  The locations of the targets will all be on the grid and not in any border cells.

## DATA PROVIDED:

There are 5 sets of Sample Data for debugging and 5 sets of Test Data for scoring. You may create additional data sets for debugging your program.

### CANDIDATE ANSWER

Language used: **Python 3**

```
#!/bin/python3

import math
import os
```

```
import random
import re
import sys



#
# Complete the 'targetsWithMostArrows' function below.
#
# The function is expected to return a STRING.
# The function accepts following parameters:
#  1. INTEGER size
#  2. STRING targets
# A: <--
#B: ^
#C: -->
#D: v
#E: Upleft
#F: Upright
#G: Downright
#H: Downleft

def checkAround(spoty, spotx, board, hitsper):
    Akg = True
    Bkg = True
    Ckg = True
    Dkg = True
    Ekg = True
    Fkg = True
    Gkg = True
    Hkg = True
    for i in range(len(board)):
        #Horizontal
        if i == 0:
            #Horizontal
            for j in range(len(board)):
                #Left
                if spotx - j > 0:
                    if board[spoty][spotx-j] != " " and Akg:
                        hitsper[str(spoty)+str(spotx-j)+"A"] = 1
                        Akg=False
                #Right
                if spotx + j < len(board)-1:
                    if board[spoty][spotx+j] != " " and Ckg:
                        hitsper[str(spoty)+str(spotx+j)+"C"] = 1
                        Ckg = False
        #Downward
        if spoty + i < len(board)-1:
            if board[spoty+i][spotx] != " " and Dkg:
                hitsper[str(spoty+i)+str(spotx)+"D"] = 1
                Dkg=False
        #Downleft
        if spoty + i < len(board)-1 and spotx - i > 0:
            if board[spoty+i][spotx-i] != " " and Hkg:
                hitsper[str(spoty+i)+str(spotx-i)+"H"] = 1
                Hkg=False
        #Downright
        if spoty + i < len(board)-1 and spotx + i < len(board)-1:
            if board[spoty+i][spotx+i] != " " and Gkg:
                hitsper[str(spoty+i)+str(spotx+i)+"G"] = 1
                Gkg=False
        #Upward
        if spoty - i > 0:
```

```
            if board[spoty-i][spotx] != " " and Bkg:
                hitsper[str(spoty-i)+str(spotx)+"B"] = 1
                Bkg=False
        #Upleft
        if spoty - i > 0 and spotx - i > 0:
            if board[spoty-i][spotx-i] != " " and Ekg:
                hitsper[str(spoty-i)+str(spotx-i)+"E"] = 1
                Ekg=False
        #Upright
        if spoty - i > 0 and spotx + i < len(board)-1:
            if board[spoty-i][spotx+i] != " " and Fkg:
                hitsper[str(spoty-i)+str(spotx+i)+"F"] = 1
                Fkg=False
    return hitsper


def targetsWithMostArrows(size, targets):
    brd = []
    targets = targets.split()
    hitsper = {}
    for o in range(len(targets)):
        for s in range(8):
            hitsper[targets[o]+chr(65+s)] = 0
    for i in range(size):
        brd.append([" "]*size)
    for n in range(size):
        for m in range(len(targets)):
            brd[int(targets[m][0])][int(targets[m][1])] = "X"
    for k in range(size):
        #top row
        hitsper = checkAround(0, k, brd, hitsper)
        #bottom row
        hitsper = checkAround(size-1, k, brd, hitsper)
        #left row
        hitsper = checkAround(k, 0, brd, hitsper)
        #right row
        hitsper = checkAround(k, size-1, brd, hitsper)
    final = []
    actual_final = ""
    for y in range(len(targets)):
        temp=0
        for z in range(8):
            temp += hitsper[targets[y]+chr(65+z)]
        final.append(temp)
    for b in range(len(targets)):
        if final[b] == max(final):
            actual_final = actual_final + targets[b] + " "
    actual_final = actual_final.split()
    actual_final.sort()
    return " ".join(actual_final)




if __name__ == '__main__':
    fptr = open(os.environ['OUTPUT_PATH'], 'w')

    size = int(input().strip())

    targets = input()

    result = targetsWithMostArrows(size, targets)
```

```
    fptr.write(result + '\n')

    fptr.close()
```

| TESTCASE | DIFFICULTY | TYPE | STATUS | SCORE | TIME TAKEN | MEMORY USED |
|---|---|---|---|---|---|---|
| Testcase 0 | Easy | Sample case | Success | 0 | 0.1068 sec | 9.57 KB |
| Testcase 1 | Easy | Sample case | Success | 0 | 0.0461 sec | 9.57 KB |
| Testcase 2 | Medium | Sample case | Success | 0 | 0.0558 sec | 9.34 KB |
| Testcase 3 | Medium | Sample case | Success | 0 | 0.0507 sec | 9.48 KB |
| Testcase 4 | Hard | Sample case | Success | 0 | 0.1034 sec | 9.44 KB |
| Testcase 5 | Easy | Hidden case | Success | 1 | 0.0886 sec | 9.56 KB |
| Testcase 6 | Medium | Hidden case | Success | 1 | 0.0719 sec | 9.45 KB |
| Testcase 7 | Medium | Hidden case | Success | 1 | 0.067 sec | 9.48 KB |
| Testcase 8 | Hard | Hidden case | Success | 1 | 0.1787 sec | 9.27 KB |
| Testcase 9 | Hard | Hidden case | Success | 1 | 0.1022 sec | 9.48 KB |

No Comments