CS 101 - PW13 - Fall 23 > aisha.batool@sse.habib.edu.pk

You can view this report online at : https://www.hackerrank.com/x/tests/1747064/candidates/57933058/report

Full Name:	Instructor
Email:	aisha.batool@sse.habib.edu.pk
Test Name:	**CS 101 - PW13 - Fall 23**
Taken On:	11 Nov 2023 13:05:54 PKT
Time Taken:	4 min 6 sec/ 10080 min
Section:	N/A
Invited by:	Aisha
Skills Score:	
Tags Score:	CS101 30/30 Lists 30/30 NestedLists 10/10 Strings 10/10

100%

50/50

scored in **CS 101 - PW13 - Fall 23** in 4 min 6 sec on 11 Nov 2023 13:05:54 PKT

Recruiter/Team Comments:

No Comments.

Plagiarism flagged

We have marked questions with suspected plagiarism below. Please review it in detail here - https://www.hackerrank.com/x/tests/1747064/candidates/57933058/report

	Question Description	Time Taken	Score	Status
Q1	Diagonal Difference > **Coding**	43 sec	10/ 10	✓
Q2	Heroes Amongst Zeroes > **Coding**	1 min 2 sec	10/ 10	✓
Q3	Check Types - Basic > **Coding**	1 min 4 sec	10/ 10	!
Q4	Special Sort > **Coding**	28 sec	10/ 10	!
Q5	Keyboard > **Coding**	29 sec	10/ 10	!
Q6	Difficulty Meter > **Multiple Choice**	4 sec	0/ 0	✓

QUESTION 1

✓

Correct Answer

Diagonal Difference > Coding

Lists NestedLists CS101

QUESTION DESCRIPTION

Score 10

Given a square matrix, calculate the absolute difference between the sums of its diagonals.

For example, the square matrix is shown below:

```
1 2 3
4 5 6
9 8 9
```

The left-to-right diagonal = 1 + 5 + 9 = 15. The right to left diagonal = 3 + 5 + 9 = 17. Their absolute difference is |15 - 17| = 2.

Function description

Complete the function in the editor below. It must return an integer representing the absolute diagonal difference.

diagonalDifference takes the following parameter:

- *arr*: an array of integers .

Input Format

The first line contains a single integer, , the number of rows and columns in the matrix .

Each of the next lines describes a row, , and consists of space-separated integers .

Constraints

- -100 <= arr[i][j] < 100

Output Format

Print the absolute difference between the sums of the matrix's two diagonals as a single integer.

Sample Input

```
3
11 2 4
4 5 6
10 8 -12
```

Sample Output

```
15
```

Explanation

The primary diagonal is:

```
11
   5
     -12
```

Sum across the primary diagonal: 11 + 5 - 12 = 4

The secondary diagonal is:

```
     4
   5
10
```

Sum across the secondary diagonal: 4 + 5 + 10 = 19

Difference: |4 - 19| = 15

Note: |x| is the absolute value of x

INTERVIEWER GUIDELINES

```
def diagonalDifference(arr):
    total1 = 0
    total2 = 0
    for i in range(len(arr)):
        total1 += arr[i][i]

    index = len(arr[0])-1
    for i in range(len(arr)):
        total2 += arr[i][index]
        index = index - 1
    return(abs(total1-total2))
```

CANDIDATE ANSWER

Language used: **Python 3**

```

def diagonalDifference(arr):
    total1 = 0
    total2 = 0
    for i in range(len(arr)):
        total1 += arr[i][i]

    index = len(arr[0])-1
    for i in range(len(arr)):
        total2 += arr[i][index]
        index = index - 1
    return(abs(total1-total2))

```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Sample case	✓ Success	1.25	0.0481 sec	9.56 KB
Testcase 1	Easy	Hidden case	✓ Success	1.75	0.0163 sec	9.52 KB
Testcase 2	Easy	Hidden case	✓ Success	1.75	0.0665 sec	9.66 KB
Testcase 3	Easy	Hidden case	✓ Success	1.75	0.0175 sec	9.59 KB
Testcase 4	Easy	Hidden case	✓ Success	1.75	0.0478 sec	9.34 KB
Testcase 5	Easy	Hidden case	✓ Success	1.75	0.0211 sec	9.6 KB

No Comments

QUESTION 2

✓ Correct Answer

Score 10

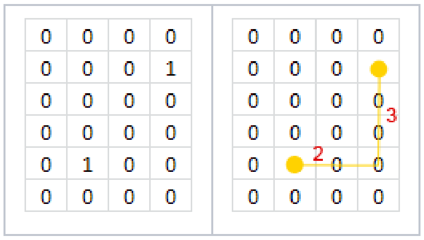
Heroes Amongst Zeroes › Coding

QUESTION DESCRIPTION

During their adventure, two heroes got lost in a grid of empty rooms and are now trying to find each other.

The rooms are represented as a grid. Each room may be empty (contains the value 0), or it may be occupied by a hero (contains the value 1). There are no other possibilities.

The *rectangular distance* between the two heroes is the shortest distance a hero must travel to reach the other, where she can only move right, left, forward, or back. In the illustration below, the rectangular distance between the heroes is 5 units.



0	0	0	0
0	0	0	1
0	0	0	0
0	0	0	0
0	1	0	0
0	0	0	0

0	0	0	0
0	0	0	●
0	0	0	0
0	0	0	0
0	●	0	0
0	0	0	0

Given such a *n x m* grid (a grid with *n* rows and *m* columns), find the rectangular distance between the two heroes.

Function Description

Write a function **rec_distance** that takes a nested list as a parameter and ***returns*** a rectangular distance between the two heroes.

Input And Output

Input and Output is handled by Hackerrank.

Examples

Input

```
6 4
0000
0001
0000
0000
0100
0000
```

Output

```
5
```

Note

The rectangular distance between the heroes is 5.

Input

```
5 5
10000
00000
00000
00000
00001
```

Output

```
8
```

Note

The rectangular distance between the heroes is 8.

Input

```
2 2
01
01
```

Output

```
1
```

Note

The rectangular distance between the heroes is 1.

INTERVIEWER GUIDELINES

```
def rec_distance(grid):
    pos = []
    for row in range(n):
        for col in range(m):
            if grid[row][col] == '1':
                pos.append((row,col))
    return abs(pos[0][0] - pos[1][0]) + abs(pos[0][1] - pos[1][1])
```

CANDIDATE ANSWER

Language used: **Python 3**

```
def rec_distance(grid):
    pos = []
    for row in range(n):
        for col in range(m):
            if grid[row][col] == '1':
                pos.append((row,col))
    return abs(pos[0][0] - pos[1][0]) + abs(pos[0][1] - pos[1][1])

```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Sample case	✓ Success	1	0.0644 sec	9.41 KB
Testcase 1	Easy	Sample case	✓ Success	1	0.0718 sec	9.63 KB
Testcase 2	Easy	Sample case	✓ Success	1	0.0251 sec	9.46 KB
Testcase 3	Easy	Sample case	✓ Success	1	0.0159 sec	9.56 KB
Testcase 4	Easy	Hidden case	✓ Success	1	0.0163 sec	9.52 KB
Testcase 5	Easy	Hidden case	✓ Success	1	0.0236 sec	9.47 KB
Testcase 6	Easy	Hidden case	✓ Success	1	0.0172 sec	9.5 KB
Testcase 7	Easy	Hidden case	✓ Success	1	0.0929 sec	9.39 KB
Testcase 8	Easy	Hidden case	✓ Success	1	0.0196 sec	9.67 KB
Testcase 9	Easy	Hidden case	✓ Success	1	0.0144 sec	9.43 KB

No Comments

QUESTION 3

!

Needs Review

Score 10

Check Types - Basic › Coding Lists CS101

QUESTION DESCRIPTION

It is often required that programs need to be checked and guarded against invalid inputs.

Write a function 'check_types' that takes as parameter a list 'lst' and returns a list of all the data types that were present in the list that was passed as a parameter. Your function should also include guards against invalid arguments.

```
>>> print(check_types([]))
[]
>>> print(check_types(["Programming", "List", "Fundamentals"]))
['str']
>>> print(check_types(['hello', [2, False, 3.5], 'world']))
['str', 'list', 'int', 'bool', 'float']
>>> print(check_types('this is not right'))
Error: Bad argument. Function 'check_types' only accept lists
```

INTERVIEWER GUIDELINES

```
def check_types(lst):
    li = []
    if type(lst) != list:
        return("Error: Bad argument. Function 'check_types' only accept
lists")
    else:
        for i in lst:
            if type(i) == int:
                if "int" not in li:
                    li.append("int")
            elif type(i) == str:
                if "str" not in li:
                    li.append("str")
            elif type(i) == list:
                if "list" not in li:
                    li.append("list")
                x = check_types(i)
                for i in x:
                    if i not in li:
                        li.append(i)
            elif type(i) == bool:
                if "bool" not in li:
                    li.append("bool")
            elif type(i) == float:
                if "float" not in li:
                    li.append("float")
    return(li)
```

CANDIDATE ANSWER

Language used: **Python 3**

```

def check_types(lst):
    li = []
    if type(lst) != list:
        return("Error: Bad argument. Function 'check_types' only accept
lists")
    else:
        for i in lst:
            if type(i) == int:
                if "int" not in li:
                    li.append("int")
            elif type(i) == str:
                if "str" not in li:
                    li.append("str")
            elif type(i) == list:
                if "list" not in li:
                    li.append("list")
                x = check_types(i)
                for i in x:
                    if i not in li:
                        li.append(i)
            elif type(i) == bool:
                if "bool" not in li:
                    li.append("bool")
            elif type(i) == float:
                if "float" not in li:
                    li.append("float")
    return(li)
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED

Testcase 0	Easy	Sample case	✓ Success	1.5	0.0193 sec	10.3 KB
Testcase 1	Easy	Hidden case	✓ Success	2	0.1469 sec	9.97 KB
Testcase 2	Easy	Sample case	✓ Success	1.5	0.0383 sec	10.3 KB
Testcase 3	Easy	Hidden case	✓ Success	2	0.0299 sec	10.3 KB
Testcase 4	Easy	Sample case	✓ Success	1.5	0.0299 sec	10 KB
Testcase 5	Easy	Sample case	✓ Success	1.5	0.0202 sec	10.3 KB

No Comments

QUESTION 4

Needs Review

Score 10

Special Sort › Coding

QUESTION DESCRIPTION

The university needs your help in sorting out its student data. The IT department has generated a list containing details of every student. However, they are having trouble sorting the data in a specific order. See if you can help your university's IT department by using your exceptional programming skills.

Write a function 'special_sort' that takes as parameter a list 'lst' and returns a sorted version of that list.

The list should be sorted according to the following order of precedence:

1. Class/Batch
2. Major
3. Name (alphabetical order)

```
>>> special_sort([['Sarwan', 'EE', '2021'], ['Lulowalokand Wala', 'CND',
'2021'], ['Hamza Junaid', 'EE', '2021'], ['Ahsan Qadeer', 'CS', '2020'],
['Muhammad Ali Bhutto', 'EE', '2020'], ['Marium Habiby', 'SDP', '2021'],
['Adil Ali Khan', 'EE', '2021']])
[['Ahsan Qadeer', 'CS', '2020'], ['Muhammad Ali Bhutto', 'EE', '2020'],
['Lulowalokand Wala', 'CND', '2021'], ['Adil Ali Khan', 'EE', '2021'],
['Hamza Junaid', 'EE', '2021'], ['Sarwan', 'EE', '2021'], ['Marium
Habiby', 'SDP', '2021']]
```

INTERVIEWER GUIDELINES

Solution

```
def special_sort(lst):
    # The list items are [name, major, batch]. The required sort order
    # is the reverse, i.e. batch, major, name. To sort according to
    # this reverse order, reverse each item in the list, sort the
    # list, and reverse each item again.
    students = lst[:]
    for student in students:
        student.reverse()
    students.sort()
    for student in students:
        student.reverse()
    return students
```

CANDIDATE ANSWER

Language used: **Python 3**

```
def special_sort(lst):
    # The list items are [name, major, batch]. The required sort order
    # is the reverse, i.e. batch, major, name. To sort according to
    # this reverse order, reverse each item in the list, sort the
    # list, and reverse each item again.
    students = lst[:]
    for student in students:
        student.reverse()
    students.sort()
    for student in students:
        student.reverse()
    return students


```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Sample case	✓ Success	2.5	0.0858 sec	9.47 KB
Testcase 1	Easy	Hidden case	✓ Success	2.5	0.0158 sec	9.41 KB
Testcase 2	Easy	Hidden case	✓ Success	2.5	0.019 sec	9.42 KB
Testcase 3	Easy	Hidden case	✓ Success	2.5	0.0251 sec	9.29 KB

No Comments

QUESTION 5

Needs Review

Score 10

Keyboard › Coding

CS101 Strings Lists

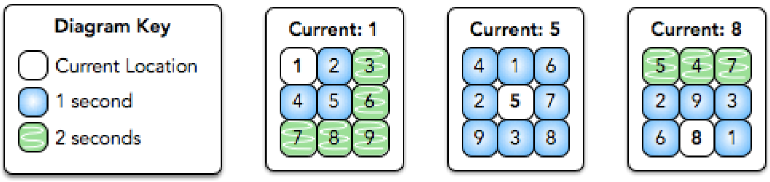
QUESTION DESCRIPTION

Challenge

You work at a secret organization where you must type a string of numbers into a console using a *3* × *3* numeric keypad. Every day they mix up the numbers on the keypad.

Use the following rules to calculate the total amount of time it takes to type a string:

- It takes *0* seconds to move your finger to the first key, and it takes *0* seconds to press the key where your finger is located any number of times.
- You can move your finger from one location to any adjacent key in one second.
- Moving to a non-adjacent key is done as a series of moves to adjacent keys.



This diagram depicts the minimum amount of time it takes to move from the current location to all other locations on the keypad.

Write a function `entryTime` that takes two parameters `s` and `keypad` and returns an integer denoting the minimum amount of time it takes to type the string `s`. keypad is a string of *9* digits where each group of *3* digits represents a row on the keypad of the day, in order.

Sample

```
>>> entryTime('423692','923857614')
8
```

Explanation

For the given sample, the keypad looks like this:

9	2	3
8	5	7
6	1	4

We calculate the time it takes to type *s* = *423692* as follows:

- *4*: We start at this number so it takes *0* seconds.
- *2*: It takes *2* seconds to move from *4* → *2*
- *3*: It takes *1* second to move from *2* → *3*
- *6*: It takes *2* seconds to move from *3* → *6*
- *9*: It takes *2* seconds to move from *6* → *9*
- *2*: It takes *1* second to move from *9* → *2*

The total time is *2* + *1* + *2* + *2* + *1* = *8*.

Input Format

The first line contains a string `s`.
The next line contains a string `keypad`.

Constraints

- `isinstance(s, str)` is `True`
- keypad is a string of *9* digits where each group of *3* digits represents a row on the keypad of the day, in order

INTERVIEWER GUIDELINES

Solution

```
def entryTime(code, keypad):
    """Returns the time taken to enter code on keypad.

    Args:
    - code (str): the code to be entered
    - keypad (str): the keys in the order of appearance on the keypad

    Observations:
    - Each letter in code must appear on keypad.
    - Entering the same key again does not take any time.
    - Moving from a position on the keypad to a neighboring position
takes 1s.
    - Moving to any other position takes 2s.

    Returns:
    int: the time taken to enter code on keypad.
    """
    # No time taken for blank code.
    if not code:
        return 0
    # Note the position on the keypad of the first key in the code. No
```

```
time is
    # taken to enter it.
    previous_position = keypad.find(code[0])
    seconds = 0
    # For each remaining key in the code, find its position on the keypad
and
    # add the time taken to move to it from the previous ley.
    for n in code[1:]:
        current_position = keypad.find(n)
        # No time taken if current key is the same as previous.
        if current_position != previous_position:
            # Neighbors add 1 second, others add 2 seconds.
            if current_position in get_neighbors(previous_position):
                seconds += 1
            else:
                seconds += 2
            # The current key becomes the previous key for the next key.
            previous_position = current_position
    return seconds


def get_neighbors(index):
    """Returns the valid neighbors of the index on the keypad.

    Args:
    - index (int): the index on keypad whose valid neighbors are
required.

    Observations:
    - index has 8 neighboring positions. Some are invalid so not saved.
    - The row and column of any idx can be obtained as divmod(idx, 3)
    - up/down neighbors are at: index -/+ 3.
    - Valid up/down neighbors have index in [0,5]/[3,8].
    - left/right neighbors are at: index -/+ 1.
    - Valid left/right neighbors are in the same row as index.
    - Valid diagonal neighbors have index in [0,8] and are in column -/+
1 as
      that of index.

    Returns:
    [int]: indexes of the valid neighbors of index.

    """
    row, col = divmod(index, 3)
    neighbors = []
    nbr = index - 3  # up
    if 0 <= nbr <= 5:
        neighbors.append(nbr)
    nbr = index + 3  # up
    if 3 <= nbr <= 8:
        neighbors.append(nbr)
    nbr = index - 1  # left
    if nbr // 3 == row:
        neighbors.append(nbr)
    nbr = index + 1  # right
    if nbr // 3 == row:
        neighbors.append(nbr)
    nbr = index - 3 - 1  # top left
    if 0 <= nbr <= 8 and nbr % 3 == col - 1:
        neighbors.append(nbr)
    nbr = index - 3 + 1  # top right
    if 0 <= nbr <= 8 and nbr % 3 == col + 1:
        neighbors.append(nbr)
    nbr = index + 3 - 1  # bottom left
    if 0 <= nbr <= 8 and nbr % 3 == col - 1:
        neighbors.append(nbr)
    nbr = index + 3 + 1  # bottom right
    if 0 <= nbr <= 8 and nbr % 3 == col + 1:
        neighbors.append(nbr)
    return neighbors
```

CANDIDATE ANSWER

Language used: **Python 3**

```python

def entryTime(code, keypad):
    """Returns the time taken to enter code on keypad.

    Args:
    - code (str): the code to be entered
    - keypad (str): the keys in the order of appearance on the keypad

    Observations:
    - Each letter in code must appear on keypad.
    - Entering the same key again does not take any time.
    - Moving from a position on the keypad to a neighboring position takes
1s.
    - Moving to any other position takes 2s.

    Returns:
    int: the time taken to enter code on keypad.
    """
    # No time taken for blank code.
    if not code:
        return 0
    # Note the position on the keypad of the first key in the code. No time
is
    # taken to enter it.
    previous_position = keypad.find(code[0])
    seconds = 0
    # For each remaining key in the code, find its position on the keypad and
    # add the time taken to move to it from the previous ley.
    for n in code[1:]:
        current_position = keypad.find(n)
        # No time taken if current key is the same as previous.
        if current_position != previous_position:
            # Neighbors add 1 second, others add 2 seconds.
            if current_position in get_neighbors(previous_position):
                seconds += 1
            else:
                seconds += 2
            # The current key becomes the previous key for the next key.
            previous_position = current_position
    return seconds


def get_neighbors(index):
    """Returns the valid neighbors of the index on the keypad.

    Args:
    - index (int): the index on keypad whose valid neighbors are required.

    Observations:
    - index has 8 neighboring positions. Some are invalid so not saved.
    - The row and column of any idx can be obtained as divmod(idx, 3)
    - up/down neighbors are at: index -/+ 3.
    - Valid up/down neighbors have index in [0,5]/[3,8].
    - left/right neighbors are at: index -/+ 1.
    - Valid left/right neighbors are in the same row as index.
    - Valid diagonal neighbors have index in [0,8] and are in column -/+ 1 as
      that of index.

    Returns:
    [int]: indexes of the valid neighbors of index.

    """
    row, col = divmod(index, 3)
    neighbors = []
    nbr = index - 3  # up
    if 0 <= nbr <= 5:
        neighbors.append(nbr)
    nbr = index + 3  # up
    if 3 <= nbr <= 8:
        neighbors.append(nbr)
    nbr = index - 1  # left
    if nbr // 3 == row:
        neighbors.append(nbr)
    nbr = index + 1  # right
    if nbr // 3 == row:
        neighbors.append(nbr)
    nbr = index - 3 - 1  # top left
    if 0 <= nbr <= 8 and nbr % 3 == col - 1:
        neighbors.append(nbr)
    nbr = index - 3 + 1  # top right
    if 0 <= nbr <= 8 and nbr % 3 == col + 1:
        neighbors.append(nbr)
    nbr = index + 3 - 1  # bottom left
    if 0 <= nbr <= 8 and nbr % 3 == col - 1:
        neighbors.append(nbr)
    nbr = index + 3 + 1  # bottom right
    if 0 <= nbr <= 8 and nbr % 3 == col + 1:
        neighbors.append(nbr)
    return neighbors
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Sample case	✓ Success	1.25	0.0201 sec	9.39 KB
Testcase 1	Easy	Sample case	✓ Success	1.25	0.0192 sec	9.46 KB
Testcase 2	Easy	Hidden case	✓ Success	1.25	0.0147 sec	9.36 KB
Testcase 3	Easy	Hidden case	✓ Success	1.25	0.0142 sec	9.39 KB
Testcase 4	Easy	Hidden case	✓ Success	1.25	0.016 sec	9.49 KB
Testcase 5	Easy	Sample case	✓ Success	1.25	0.0163 sec	9.59 KB
Testcase 6	Easy	Sample case	✓ Success	1.25	0.0143 sec	9.38 KB
Testcase 7	Easy	Hidden case	✓ Success	1.25	0.0615 sec	9.51 KB

No Comments

QUESTION 6

✓ Correct Answer

Score 0

Difficulty Meter › Multiple Choice

QUESTION DESCRIPTION

On a scale of 1 to 5, with 1 being very easy and 5 being extremely challenging, how would you rate this worksheet?

CANDIDATE ANSWER

Options: (Expected answer indicated with a tick)

✓ ● 1
✓ ○ 2
✓ ○ 3
✓ ○ 4
✓ ○ 5

No Comments

PDF generated at: 11 Nov 2023 08:12:09 UTC