You can view this report online at : https://www.hackerrank.com/x/tests/1508803/candidates/49324656/report

Full Name:	Breeha Qasim
Email:	bq08283@st.habib.edu.pk
Test Name:	**CS102 - Lab 2 - Spring 2023**
Taken On:	20 Jan 2023 10:34:27 PKT
Time Taken:	2274 min 8 sec/ 2880 min
Work Experience:	< 1 years
Invited by:	Aisha
Skills Score:	
Tags Score:	CS102 100/100 Hard 100/100 NestedLists 100/100

100%
400/400

scored in **CS102 - Lab 2 - Spring 2023** in 2274 min 8 sec on 20 Jan 2023 10:34:27 PKT

Recruiter/Team Comments:

No Comments.

Plagiarism flagged

We have marked questions with suspected plagiarism below. Please review.

	Question Description	Time Taken	Score	Status
Q1	Matrix Subtraction > **Coding**	10 min 54 sec	100/ 100	!
Q2	Transpose of a Matrix > **Coding**	13 min 27 sec	100/ 100	✓
Q3	Matrix Multiplication > **Coding**	36 min 31 sec	100/ 100	✓
Q4	Image Sharpening > **Coding**	1 hour 10 min 4 sec	100/ 100	✓

QUESTION 1

Needs Review

Score 100

Matrix Subtraction > Coding

QUESTION DESCRIPTION

Challenge

Write a function, `matrix_subtraction`, that takes two integer matrices, `A` and `B`, and returns a matrix whose entries are the subtraction of the corresponding entries in matrix `B` from A.

Sample

```
>>> matrix_subtraction( [[1,2,3],[4,5,6],[7,8,9]],[[9,8,7],[6,5,4],
[3,2,1]] )
[[-8, -6, -4],[-2, 0, 2],[4, 6, 8]]
>>> matrix_subtraction([[12,7,3],[4,5,6],[7,8,9]],[[5,8,1],[6,7,3],
[4,5,9]] )
[[7,-1,2],[-2,-2,3],[3,3,0]]
>>> matrix_subtraction([[1],[1],[1]],[[2],[2],[4]])
[[-1],[-1],[-3]]
>> matrix_subtraction([[1],[2]],[[3,5],[4,6]])
Matrices A and B don't have the same dimension required for matrix
subtraction.
```

INTERVIEWER GUIDELINES

```
def matrix_subtraction(X, Y):
    # Matrix X MINUS Matrix Y
    # return the resulting matrix

    size1 = (len(X), len(X[0]))
    size2 = (len(Y), len(Y[0]))

    if size1 != size2:
        return("Matrices A and B don't have the same dimension required
for matrix subtraction.")

    Z = []

    for i in range(len(X)):
        list = []
        for j in range(len(X[i])):
            list.append(X[i][j] - Y[i][j])
        Z.append(list)

    return Z
```

CANDIDATE ANSWER

Language used: **Python 3**

```
def matrix_subtraction(A,B):
    x=len(A)
    y=len(B)
    final_lst=[]
    if x==y:
        for i in range(x):
            if len(A[i])!=len(B[i]):
                return "Matrices A and B don't have the same dimension
required for matrix subtraction."
            lst=[]
            for j in range(len(A[i])):
                lst.append(A[i][j]-B[i][j])
            final_lst.append(lst)
        return final_lst
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Sample case	✓ Success	10	0.1106 sec	8.64 KB

Testcase 1	Easy	Sample case	✓ Success	10	0.079 sec	9.02 KB
Testcase 2	Easy	Sample case	✓ Success	10	0.0644 sec	9.04 KB
Testcase 3	Easy	Hidden case	✓ Success	20	0.0525 sec	8.99 KB
Testcase 4	Easy	Hidden case	✓ Success	20	0.087 sec	9.02 KB
Testcase 5	Easy	Sample case	✓ Success	10	0.079 sec	9.02 KB
Testcase 6	Easy	Hidden case	✓ Success	20	0.0713 sec	8.93 KB

No Comments

QUESTION 2

✓ Correct Answer

Score 100

Transpose of a Matrix › Coding

QUESTION DESCRIPTION

Challenge

The transpose of a matrix is a new matrix whose rows are the columns of the original.

Write a function, `matrix_transpose`, that takes an integer matrix, `A`, and returns its transpose.

Sample

```
>>>matrix_transpose([[12,7],[4 ,5],[3 ,8]])
[[12, 4, 3],[7, 5, 8]]
>>>matrix_transpose([[12, 4, 3],[7, 5, 8]])
[[12,7],[4 ,5],[3 ,8]]
```

INTERVIEWER GUIDELINES

```
def matrix_transpose(X):
    # Transpose Matrix X
    # Return the resulting matrix
    Z = []

    columns = len(X[0])

    for i in range(len(X[0])):
        list = []
        for j in range(len(X)):
            list.append(X[j][i])
        Z.append(list)

    return Z
```

CANDIDATE ANSWER

Language used: **Python 3**

```
def matrix_transpose(A):
    final=[]
    for i in range(len(A[0])):
        lst=[]
        for x in range(len(A)):
            lst.append(A[x][i])
        final.append(lst)
    return final

```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Sample case	✓ Success	20	0.0428 sec	8.85 KB
Testcase 1	Easy	Sample case	✓ Success	20	0.0903 sec	8.87 KB
Testcase 2	Easy	Hidden case	✓ Success	20	0.0577 sec	9.1 KB
Testcase 3	Easy	Hidden case	✓ Success	20	0.0836 sec	9.01 KB
Testcase 4	Easy	Hidden case	✓ Success	20	0.053 sec	8.73 KB

No Comments

QUESTION 3

✓ Correct Answer

Score 100

Matrix Multiplication › Coding

QUESTION DESCRIPTION

Challenge

Write a function, `matrix_multiplication`, that takes two integer matrices, `A` and `B`, and returns their dot product.

Reference: https://www.mathsisfun.com/algebra/matrix-multiplying.html

Sample

```
>>> matrix_multiplication([[12,7,3],[4 ,5,6],[7 ,8,9]], [[5,8,1,2],
[6,7,3,0], [4,5,9,1]])
[[114, 160, 60, 27],[74, 97, 73, 14],[119, 157, 112, 23]]
>>> matrix_multiplication([[34,1,77],[2,14,8],[3 ,17,11]], [[6,8,1],
[9,27,5],[2,43,31]])
[[367, 3610, 2426], [154, 738, 320], [193, 956, 429]]
>>> matrix_multiplication([[1,2,3],[4,5,6]], [[7,8],[9,10],[11,12]])
[[58, 64], [139, 154]]
>>> matrix_multiplication([[7,3], [2,5], [6,8], [9,0]], [[8,14,0,3,-1],
[7,11,5,91,3], [8,-4,19,5, 57]])
The number of columns in Matrix A does not equal the number of rows in
Matrix B required for Matrix Multiplication.
```

INTERVIEWER GUIDELINES

```
def matrix_multiplication(X, Y):
    # Multiply matrices X and Y
    # Return the resulting matrix
    Z = []

    size1 = (len(X), len(X[0]))
    size2 = (len(Y), len(Y[0]))

    if size1[1] != size2[0]:
        return("The number of columns in Matrix A does not equal the
number of rows in Matrix B required for Matrix Multiplication.")
```

```
    for i in range(len(X)):
        list = []
        for j in range(len(Y[0])):
            num = 0
            for k in range(len(Y)):
               num += (X[i][k] * Y[k][j])
            list.append(num)
        Z.append(list)

    return Z
```

CANDIDATE ANSWER

Language used: **Python 3**

```
def matrix_multiplication(A,B):
    lst=[]
    len_A=len(A)
    len_B=len(B[0])
    if len(A[0])!=len(B):
        return "The number of columns in Matrix A does not equal the number of rows in Matrix B required for Matrix Multiplication."
    #lst2=[]
    for i in range(len_A):
        lst1=[]
        for j in range(len_B):
            lst1.append(0)
        lst.append(lst1)
    for i in range(len_A):
        for j in range(len_B):
            for k in range(len(B)):
                lst[i][j]+=A[i][k]*B[k][j]
    return lst
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Sample case	✓ Success	10	0.0586 sec	8.87 KB
Testcase 1	Easy	Sample case	✓ Success	10	0.0889 sec	8.77 KB
Testcase 2	Easy	Sample case	✓ Success	10	0.0544 sec	8.98 KB
Testcase 3	Easy	Sample case	✓ Success	10	0.0644 sec	8.93 KB
Testcase 4	Easy	Sample case	✓ Success	10	0.0777 sec	8.92 KB
Testcase 5	Easy	Sample case	✓ Success	10	0.0686 sec	8.83 KB
Testcase 6	Easy	Sample case	✓ Success	10	0.0752 sec	8.76 KB
Testcase 8	Easy	Sample case	✓ Success	10	0.0542 sec	8.82 KB
Testcase 9	Easy	Sample case	✓ Success	10	0.0496 sec	9.02 KB
Testcase 10	Easy	Sample case	✓ Success	10	0.0556 sec	8.97 KB

No Comments

QUESTION 4

Image Sharpening › Coding Hard CS102 NestedLists

Correct Answer

Score 100

QUESTION DESCRIPTION

Challenge

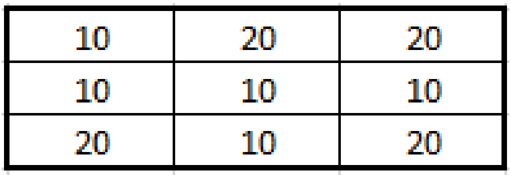
An image can be sharpened by multiplying every pixel by 2, and then subtracting the average value of the neighborhood(up,down,left,right) from it. The resultant pixel value would be an absolute value.

Write a function `sharpen_image()` that takes as parameter an image in the form of a nested list `A` and sharpens it.

For example:

Input:

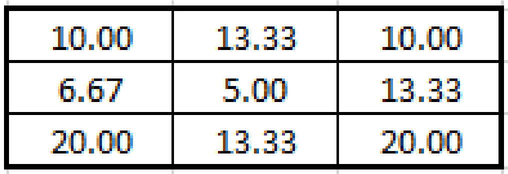
B =



10	20	20
10	10	10
20	10	20

Output:

C =



10.00	13.33	10.00
6.67	5.00	13.33
20.00	13.33	20.00

C_{00} = abs ((B_{00} * 2) - ((B_{01} * 2) + (B_{10} * 2)) / 2)
= abs (20 - (40 + 20) / 2)
= abs (20 - 30)
= 10

C_{11} = abs ((B_{11} * 2) - ((B_{10} * 2) + (B_{01} * 2) + (B_{12} * 2) + (B_{21} * 2) / 4)
= abs (20 - (20 + 40 + 20 + 20) / 4)
= abs (20 - 25)
= 5

Note : Not all of the neighbors are available in boundary cases. You have to write suitable conditions accordingly.
Also, neighbors of a pixel are top, bottom, left and right pixels.
Also, All pixel values are rounded off to two decimal places.

INTERVIEWER GUIDELINES

```
def sharpen_image(lst):
    for i in range(len(lst)):
        for j in range(len(lst[i])):
            lst[i][j] = (lst[i][j]) * 2

    B=[]

    for i in range(len(lst)):
        R=[]
        for j in range(len(lst[i])):
            R.append(0)
        B.append(R)

    for i in range(len(lst)):
        for j in range(len(lst[i])):
            if i == 0:
                if j == 0:
                    B[i][j] = abs(lst[i][j] - (lst[i][j+1] + lst[i+1]
[j])/2)
                elif j == len(lst[i])-1:
                    B[i][j] = abs(lst[i][j] - (lst[i][j-1] + lst[i+1]
[j])/2)
                else:
                    B[i][j] = abs(lst[i][j] - (lst[i][j-1] + lst[i]
[j+1]+lst[i+1][j])/3)
            elif i == len(lst) - 1:
                if j == 0:
                    B[i][j] = abs(lst[i][j] - (lst[i][j+1] + lst[i-1]
[j])/2)
                elif j == len(lst[i])-1:
                    B[i][j] = abs(lst[i][j] - (lst[i][j-1] + lst[i-1]
[j])/2)
                else:
                    B[i][j] = abs(lst[i][j] - (lst[i][j-1] + lst[i]
[j+1]+lst[i-1][j])/3)
            elif j == 0:
                B[i][j] = abs(lst[i][j] - (lst[i-1][j] + lst[i][j+1] +
lst[i+1][j])/3)
            elif j == len(lst[i]) - 1:
                B[i][j] = abs(lst[i][j] - (lst[i-1][j] + lst[i][j-1] +
lst[i+1][j])/3)
            else:
                B[i][j] = abs(lst[i][j] - (lst[i][j-1] + lst[i-1][j] +
lst[i][j+1] + lst[i+1][j])/4)

            B[i][j] = round(B[i][j], 2)

    return B
```

```
def in_range(index, I):
    if ((index[0] >= 0 and index[0] < len(I)) and ((index[1] >= 0 and
index[1] < len(I[0])))):
        return True
    return False

def sharpen_image(I):
    NI=[]
    for i in range(len(I)):
        nrow=[]
        for j in range(len(I[0])):

            pixel_value_total =0
            n=0

            if(in_range((i-1, j), I)):          #up
                pixel_value_total += I[i-1][j]*2
                n+=1
            if(in_range((i+1, j), I)):          #down
                pixel_value_total += I[i+1][j]*2
                n+=1
            if(in_range((i, j-1), I)):          #left
                pixel_value_total += I[i][j-1]*2
                n+=1
            if(in_range((i, j+1), I)):          #right
                pixel_value_total += I[i][j+1]*2
                n+=1


            val = round(abs(I[i][j] * 2 - pixel_value_total/n),2)

            nrow.append(val)

        NI.append(nrow)

    return NI
```

CANDIDATE ANSWER

Language used: **Python 3**

```python
def sharpen_image(A):
    lst=[]
    for x in range(len(A)):
        lst2=[]
        for i in range(len(A[0])):
            down=0
            up=0
            left=0
            right=0
            if x-1>=0:
                up=A[x-1][i]
            if x+1<len(A):
                down=A[x+1][i]
            if i+1<len(A[0]):
                right=A[x][i+1]
            if i-1>=0:
                left=A[x][i-1]
            counter=0
            if left!=0:
                counter=counter+1
            if right!=0:
                counter=counter+1
            if up!=0:
                counter=counter+1
            if down!=0:
                counter=counter+1
            lst2.append(round(abs((A[x][i]*2)-((left*2)+(right*2)+(up*2)+
(down*2))/counter),2))
        lst.append(lst2)
    return lst
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Sample case	✓ Success	10	0.0501 sec	9.01 KB
Testcase 1	Easy	Hidden case	✓ Success	15	0.0503 sec	9.18 KB
Testcase 2	Easy	Sample case	✓ Success	10	0.0582 sec	8.95 KB
Testcase 3	Easy	Hidden case	✓ Success	15	0.0877 sec	9.03 KB
Testcase 4	Easy	Hidden case	✓ Success	20	0.0534 sec	9.23 KB
Testcase 5	Easy	Sample case	✓ Success	10	0.0508 sec	8.99 KB
Testcase 6	Easy	Hidden case	✓ Success	20	0.0551 sec	8.96 KB

No Comments

No Comments