Solution Lab 07
OOP – BSDS

Today's Lab will held on Hacker Rank. I have already given instructions to create logins and attempt practice sessions. I am pasting problems here, but you need to attempt them on hacker rank. The link is:
www.hackerrank.com/lab7-oop-spring-2022

S_Most_Economing

Three persons purchased different things in the market. Find who spent more economically. In input purchase information of three persons are given one by one. For each there is count of things purchased, followed by amount spent on each thing. The most economically means average price is minimum. See input, output for further understanding:

Input Format

3 250 750 500
6 150 100 200 240 130 170
2 200 300

Constraints

n>0

Output Format

Person 2 spent more economically

```
def read_data_and_return_average():
    s = input()
    s_values=s.split()
    sum = 0
    n = int(s_values[0])
    for i in range(1, n+1):
        sum += int(s_values[i])
    return sum/n

def main():
    best = 1
    best_avg = read_data_and_return_average()
    curr_avg = read_data_and_return_average()
    if best_avg > curr_avg:
        best = 2
        best_avg = curr_avg
    curr_avg = read_data_and_return_average()
    if best_avg > curr_avg:
        best = 3
        best_avg = curr_avg
    print (f'Person {best} spent more economically')

main()
```

M_Total_IQ

ABC Company has 24 employees sitting in a hall. There are four rows and six employees sitting in each row. They are working in team of 2x2 four employees. Each employee has different IQ. The minimum IQ in the team is considered as overall IQ of the team. Your task is to find total IQ of all the teams. Consider input format, IQ of members of first team is 7, 9, 6 & 3. The minimum IQ is 3, hence 3 is considered IQ of team 1. Similarly, team 2 has IQ 4 that is minimum among 8, 6, 5, and 4. In this way IQ of six teams is respectively 3, 4, 2, 2, 3, 5. Finally, total IQ of teams is 19, by summing 3+4+2+2+3+5.

Input Format

7 9 8 6 9 5
6 3 5 4 2 7
5 8 3 4 9 5

2 5 7 8 7 4

Constraints

IQ>0

Output Format

19

```
def read_data():
    iq = [[] for i in range(4)]
    for i in range(4):
        s = input()
        s_values=s.split()
        for element in s_values:
            iq[i].append(int(element))
    return iq

def find_total_iq(iq):
    sum = 0
    for i in range(0, 4, 2):
        for j in range(0, 6, 2):
            sum += min(iq[i][j],iq[i][j+1],iq[i+1][j],iq[i+1][j+1])
    return sum
def main():
    iq = read_data()
    print (find_total_iq(iq))

main()
```

M_Weighted_Average_R

Find weighted average price of items having price within the range. In input minimum and maximum range of price is 50 & 100. The company has purchased five items, given in second line of input. In last line pair of quantity and price is given. For example, first pair is 60, 70. 60 is quantity of first item & 70 is price of first item.

Price of second item is 120, larger than range and price of second last item is 30, smaller than range. Exclude second & second last item. Amount of total purchase: 60 x 70 + 90 x 80 + 40 x 60 = 13800 Total units of all items: 60 + 90 + 40 = 190 weighted average:13800/190= 73 (rounded)

Input Format

50 100

5

60 70 95 120 90 80 95 30 40 60

Constraints

Price>0

Items>0

Type of items>0

Output Format

13800 190 73

```
def read_min_max():
    val_str = input().split()
    return int(val_str[0]), int(val_str[1])

def read_data_print_output(n, min, max):
    val_str = input().split()
    quantity =[int(val_str[i]) for i in range(0, len(val_str), 2)]
    price =[int(val_str[i]) for i in range(1, len(val_str), 2)]
    total_purchase = 0
    total_units = 0
```

```
        for i in range(len(price)):
            if price[i] > min and price[i] < max:
                total_purchase += price[i] * quantity[i]
                total_units += quantity[i]
        print (f'{total_purchase} {total_units}
    {round(total_purchase/total_units,0):.0f}')


    def main():
        min, max = read_min_max()
        n = int(input())
        read_data_print_output (n, min, max)

    main()
```

M_Mirror_Match

Read a square matrix of 5x5. Do a mirror match of elements in the matrix and place 0 if both elements are same, otherwise keep elements at their place. Finally, print the mirror map. Here mirror map means compare first row with last row, second row with second last row, and middle row with middle row. Similarly, match first column with last column, second column with second last column and third column with third column.

Input Format

7 9 8 6 2
6 3 5 4 3
5 8 3 4 5
2 4 7 8 6
2 3 8 9 4

Constraints

only 5x5 matrix

Output Format

7 0 0 6 0
0 3 5 0 3
0 8 0 4 0
2 0 7 8 0
0 3 0 0 4

Sample Input 0

7 9 8 6 2
6 3 5 4 3
5 8 3 4 5
2 4 7 8 6
2 3 8 9 4

Sample Output 0

7 0 0 6 0
0 3 5 0 3
0 8 0 4 0
2 0 7 8 0
0 3 0 0 4

```
    def read_data():
        val = [[] for i in range(5)]
        for i in range(5):
            s = input()
            s_values=s.split()
            for element in s_values:
                val[i].append(int(element))
        return val
```

```
def mirror(val):
    n = len(val)
    for i in range(3):
        for j in range(5):
            if val[i][j] == val[n-i-1][n-j-1]:
                val[i][j] = 0
                val[n-i-1][n-j-1] = 0
    for elements in val:
        for element in elements:
            print (element, end=' ')
        print()

def main():
    val = read_data()
    mirror(val)

main()
```