Week-5, Graded, Programming

Week-5, Graded, Programming
- Problem 1
  - Question
  - Answer
  - Suffix Code Block
  - Test Cases
    - Public
    - Private
- Problem 2
  - Question
  - Suffix Code Block
  - Answer
  - Test Cases
    - Public
    - Private
- Problem 3
  - Question
  - Suffix Code Block
  - Answer
  - Test Cases
    - Public
    - Private
- Problem-4
  - Question
  - Test Cases
  - Answer
  - Suffix Code Block
- Problem-5
  - Question
  - Test Cases
  - Answer
  - Suffix Code Block

Problem 1

Question

A perfect number is a positive integer that is equal to the sum of its proper positive divisors (excluding itself). Fill the body of the function `perfect_number` to check whether an input integer is perfect or not. It should return boolean literal `True` if the number is perfect. Otherwise, it should return `False`.

```python
def perfect_number(num):
    # Write function body
    #
    #
    #

```

Note

- You don't need to accept the input from the user or print the output to the console. This will be processed internally.
- You only need to fill the details in the body of the function.

Answer

```python
'''
A perfect number is a positive integer that is equal to the sum of its proper positive divisors. That is, the sum of its positive divisors excluding the number itself (also known as its aliquot sum).
'''
def perfect_number(num):
    sum = 0
    if num < 2:
        return False
    else:
        for x in range(1, num):
            if num % x == 0:
                sum += x
        return sum == num
```

Suffix Code Block

```python
print(perfect_number( int(input()) ))
```

Test Cases

Public

Input	Factor	Output
8	1, 2, 4	`False`
28	1, 2, 4, 7, 14	`True`

Private

Input	Factor	Output
496	1, 2, 3	`True`
8128	1, 2, 4, 7, 14	`True`
1	1	`False`
-1	-1	`False`

Problem 2

Question

You are a forum admin. You wish to find out the total score of each user using the scoring logic given below. If the user scores more than `50`, then the user is given a `Leader` badge, otherwise a `Basic` badge is given.

Action	Points
`read a post`	1
`replied to a post`	3
`created a new post`	5

Implement the body of the function `user_score`. The function accepts 3 integer arguments — `read_count, reply_count, new_post_count`. It calculates the score based on the input counts and returns `Leader` or `Basic`.

```python
def user_score(read_count, reply_count, new_post_count):
    # Write function body
    #
    #
    #
```

Note

- You don't need to accept the input from the user or print the output to the console. This will be processed internally.
- You only need to fill the details in the body of the function.

Suffix Code Block

```python
read_count, reply_count, new_post_count = input().split(",")
print(user_score( int(read_count), int(reply_count), int(new_post_count) ))
```

Answer

```python
def user_score(read_count, reply_count, new_post_count):
    score =  read_count * 1 + reply_count * 3 + new_post_count * 5
    if score > 50:
        return "Leader"
    else:
        return "Basic"
```

Test Cases

Public

Input	Output
`1,1,1`	`Basic`
`10,10,10`	`Leader`

Private

Input	Output
`0,0,0`	`Basic`
`50,0,0`	`Basic`
`0,16,0`	`Basic`
`0,0,10`	`Basic`
`51,0,0`	`Leader`
`0,17,0`	`Leader`
`1,0,10`	`Leader`

Problem 3

Question

In the Gregorian calendar, a leap year has a total of 366 days instead of the usual 365 as a result of adding an extra day (February 29) to the year. This calendar was introduced in 1582, to replace the flawed Julian Calendar. Below criteria are used to determine if a year is a leap year or not.

- If a year is divisible by `100` then it will be a leap year if it is also divisible by `400`
- If a year is not divisible by `100`, then it will be a leap year if it is divisible by `4`.

Implement the body of the function `check_leap_year` to incorporate above logic. The function takes any year from 1600 to 9999 (endpoints included) as an input number and should return Boolean value `True` when the year is a leap year, otherwise it should return `False` .

```python
def check_leap_year(year):
    # Write function body
    #
    #
    #
```

Suffix Code Block

```python
print(check_leap_year( int(input()) ))
```

Answer

```python
def check_leap_year(year):
    if (year % 400 == 0) or (year % 100 != 0 and year % 4 == 0):
        return True
    else:
        return False
```

Test Cases

Public

Input	Output
2020	`True`
2021	`False`

Private

Input	Output
1900	`False`
2100	`False`
2000	`True`
9999	`False`

Problem-4

Question

A $n \times n$ square matrix of positive integers is called a magic square if the following sums are equal:

- row-sum: sum of numbers in every row; there are n such values, one for each row
- column-sum: sum of numbers in every column; there are n such values, one for each column
- diagonal-sum: sum of numbers in both the main diagonals; there are two values

There are $n + n + 2 = 2n + 2$ values involved. All these values must be the same for the matrix to be a magic-square.

Task

Write a function `is_magic` that accepts a matrix as input and returns `YES` if it is a magic-square and `NO` if it isn't one. You can assume the following:

- The first line of input will have the dimension n of the matrix.
- The next n lines will be a sequence of n space-separated integers.

```python
def is_magic(mat):
    """Magic square!"""
    pass
```

Note

- You don't need to accept the input from the user or print the output to the console. This will be processed internally.
- You only need to fill the details in the body of the function.

Test Cases

Type	Input	Output

Public	`2`
	`1 2`
	`2 1`
		NO

Public	`3`
	`4 9 2`
	`3 5 7`
	`8 1 6`
		YES

Private	`4`
	`2 16 13 3`
	`11 5 8 10`
	`7 9 12 6`
	`14 4 1 15`
		YES

Private	`3`
	`1 2 3`
	`4 5 6`
	`7 8 9`
		NO

Private	`2`
	`1 1`
	`1 1`
		YES

Answer

```python
def is_magic(mat):
    # first get the dimension of the matrix
    m = len(mat)
    # the sum of the two diagonals
    d1sum, d2sum = 0, 0
    # (i, i) goes from top-left -> bottom-right
    # (i, m - i - 1) goes from top-right -> bottom-left
    # note that a single loop is enough; no nesting required
    for i in range(m):
        d1sum += mat[i][i]
        d2sum += mat[i][m - i - 1]
    # if the two diagonal sums are unequal, we can return NO
    # unnecessary computation can be avoided
    if not(d1sum == d2sum):
        return 'NO'
    # get row-sum and column-sum
    for i in range(m):
        rsum, csum = 0, 0
        for j in range(m):
            rsum += mat[i][j]
            csum += mat[j][i]
        if not(rsum == csum == d1sum):
            return 'NO'
    # if the code reaches this level
    # then all requirements of a magic-square are satisfied
    # so we can safely return YES
    return 'YES'
```

Suffix Code Block

```
m = int(input())
mat = [ ]
for i in range(m):
    mat.append([ ])
    for num in input().split():
        mat[i].append(int(num))

print(is_magic(mat))
```

Problem-5

Question

n integers are written on a blackboard in a classroom. Consider the following process:

- Head to the board and pick the two smallest integers from it.
- Make a note of them in your head and then erase the two numbers from the board.
- Find the absolute value of their difference and write the result on the board.

Repeat the whole process again. Keep doing it until only one number is left on the board. Write this number down on a piece of paper and erase the board. If there is only one number to begin with, all other operations can be avoided. You just have to record this number on paper and then erase it from the board.

Assumptions

- There will be at least one number on the board to begin with.
- The board should be empty after the process is complete. No numbers must be written on it.

Task

The following functions are given to you and you don't have to write them:

- `board_isEmpty()`: returns `True` if the board is empty and `False` otherwise
- `board_write(num)`: accepts `num` as an argument and writes the `num` on the board
- `board_erase(num)`: accepts `num` as an argument and erases the `num` from the board
- `board_min()`: returns the minimum number on the board; doesn't erase it from the board

Your task is to write a function named `process` that will execute this process given in the problem statement using the functions given above and return the number on the piece of paper.

```python
def process():
    # Execute the process #
    # Return the number on the piece of paper #
    pass
```

Input-Output

Each test case corresponds to some collection of numbers on the board. The actual input (numbers on the board) will be hidden from your view. Instead the input column in each test case will contain the test-case number.

The output will be the number you have written down in the piece of paper corresponding to that test case. Remember, we will also be checking if the board has been wiped clean at the end!

Hint

The first public test case has the following integers written on the board:

> 1 2 3 4 5

Numbers on the board at the beginning of each iteration of the process:

```
1,2,3,4,5
1,3,4,5
2,4,5
2,5
3

```

Test Cases

Type	Input	Output
Public	1	3
Public	2	4
Private	3	100
Private	4	30
Private	5	28

Answer

```python
def process():
    # There will be at least one number on the board
    while True:
        # Extract the minimum number
        n1 = board_min()
        # Erase it from the board
        board_erase(n1)
        # If the board is empty; then our job is done
        if board_isEmpty():
            return(n1)
            break
        # As the board is not empty, there is one more number
        n2 = board_min()
        # Erase it
        board_erase(n2)
        # Find the difference
        n3 = abs(n1 - n2)
        # Write it on the board
        board_write(n3)
        # Loop back
```

Suffix Code Block

```python
## Prefix Code Block ##
def board_isEmpty():
    return l == [ ]

def board_write(num):
    l.append(num)

```

```python
def board_erase(num):
    l.remove(num)

def board_min():
    return min(l)

test_cases = [ [1, 2, 3, 4, 5],
               [2, 4, 6, 8],
               [100],
               [120, 90],
               [10, 30, -4, -90, 100, 39, -4, 1, 10]
             ]

test_case_id = int(input()) - 1
l = test_cases[test_case_id]
process()
assert l == [ ]
```