# Rajalakshmi Engineering College

Name: Rena J
Email: 241801227@rajalakshmi.edu.in
Roll no: 241801227
Phone: 9941271176
Branch: REC
Department: I AI & DS FC
Batch: 2028
Degree: B.E - AI & DS

Scan to verify results

## NeoColab_REC_CS23221_Python Programming

## REC_Python_Week 2_COD_Updated

Attempt : 1
Total Mark : 50
Marks Obtained : 50

### Section 1 : Coding

1. Problem Statement

John, a software developer, is analyzing a sequence of numbers within a given range to calculate their digit sum. However, to simplify his task, he excludes all numbers that are palindromes (numbers that read the same backward as forward).

Help John find the total sum of the digits of non-palindromic numbers in the range [start, end] (both inclusive).

Example:

Input:

10

20

Output:
55

Explanation:

Range [10, 20]: Non-palindromic numbers are 10, 12, 13, 14, 15, 16, 17, 18, 19 and 20.

Digit sums: 1+0 + 1+2 + 1+3 + 1+4 + 1+5 + 1+6 + 1+7 + 1+8 + 1+9 + 2+0 = 55.

Output: 55

***Input Format***

The first line of input consists of an integer, representing the starting number of the range.

The second line of input consists of an integer, representing the ending number of the range.

***Output Format***

The output prints a single integer, representing the total sum of the digits of all non-palindromic numbers in the range.

Refer to the sample output for formatting specifications.

***Sample Test Case***

Input: 10
20
Output: 55

***Answer***

```
# You are using Python
def palindrome(n):
    return str(n)==str(n)[::-1]

def sumofdigit(n):
    return sum(int(d) for d in str(n))
```

```
def non_palindrome(start,end):
    total_sum =0;
    for num in range(start,end+1):
        if not palindrome(num):
            total_sum +=sumofdigit(num)
    return total_sum

start=int(input())
end=int(input())
print(non_palindrome(start,end))
```

***Status :*** Correct ***Marks : 10/10***

2. Problem Statement

As a junior developer working on a text analysis project, your task is to create a program that displays the consonants in a sentence provided by the user, separated by spaces.

You need to implement a program that takes a sentence as input and prints the consonants while skipping vowels and non-alphabetic characters using only control statements.

***Input Format***

The input consists of a string representing the sentence.

***Output Format***

The output displays space-separated consonants present in the sentence.

Refer to the sample output for the formatting specifications.

***Sample Test Case***

Input: Hello World!
Output: H l l W r l d

***Answer***

```
# You are using Python
def consonants(sentence):
    vowels="aeiouAEIOU"
    result=[]
    for char in sentence:
        if char.isalpha() and char not in vowels:
            result.append(char)
    print(" ".join(result))

sentence=input()
consonants(sentence)
```

***Status :*** Correct ***Marks : 10/10***

3. Problem Statement

You work as an instructor at a math enrichment program, and your goal is to develop a program that showcases the concept of using control statements to manipulate loops. Your task is to create a program that takes an integer 'n' as input and prints the squares of even numbers from 1 to 'n', while skipping odd numbers.

***Input Format***

The input consists of a single integer, which represents the upper limit of the range.

***Output Format***

The output displays the square of even numbers from 1 to 'n' separated by lines.

Refer to the sample output for the formatting specifications.

***Sample Test Case***

Input: 10

Output: 4
16
36
64

100

***Answer***

```
# You are using Python
def even_sq(n):
    for i in range(1,n+1):
        if i%2 !=0:
            continue
        print(i**2)

n=int(input())
even_sq(n)
```

***Status :*** Correct ***Marks : 10/10***

4. Problem Statement

Emma, a mathematics enthusiast, is exploring a range of numbers and wants to count how many of them are not Fibonacci numbers.

Help Emma determine the count of non-Fibonacci numbers within the given range [start, end] using the continue statement.

***Input Format***

The first line of input consists of an integer, representing the starting number of the range.

The second line consists of an integer, representing the ending number of the range.

***Output Format***

The output prints a single integer, representing the count of numbers in the range that are not Fibonacci numbers.

Refer to the sample output for formatting specifications.

***Sample Test Case***

Input: 1
10
Output: 5

***Answer***

```
# You are using Python
import math
def fibonacci(num):
    if num<0:
        return False
    x1=5*num*num+4
    x2=5*num*num-4
    return math.isqrt(x1)**2==x1 or math.isqrt(x2)**2 ==x2

def count_non_fibo(start,end):
    count=0
    for num in range(start,end+1):
        if fibonacci(num):
            continue
        count+=1
    return count

start=int(input())
end=int(input())
print(count_non_fibo(start,end))
```

***Status :*** Correct ***Marks : 10/10***

5. Problem Statement

Ethan, a curious mathematician, is fascinated by perfect numbers. A perfect number is a number that equals the sum of its proper divisors (excluding itself). Ethan wants to identify all perfect numbers within a given range.

Help him write a program to list these numbers.

***Input Format***

The first line of input consists of an integer start, representing the starting number of the range.

The second line consists of an integer end, representing the ending number of the range.

***Output Format***

The output prints all perfect numbers in the range, separated by a space.

Refer to the sample output for formatting specifications.

***Sample Test Case***

Input: 1
100
Output: 6 28

***Answer***

```
# You are using Python
def is_perfect_number(num):
    if num<2:
        return False
    divisor_sum=sum(i for i in range(1,num)if num%i==0)
    return divisor_sum==num

def find_perfect(start,end):
    perfect_number =[]
    for num in range(start , end+1):
        if is_perfect_number(num):
            perfect_number.append(str(num))

    print(" ".join(perfect_number))

start =int(input())
end = int(input())
find_perfect(start,end)
```

***Status :*** Correct ***Marks : 10/10***