**04 – Iteration control structures**

**Ex. No. : 4.1 Date: 02/05/2024**

**Register No.: 231401013 Name: ARCHITHA. M**

**Count of Unique Digits**

Write a program to find the count of unique digits in a given number N. The number will be passed to the program as an input of type int. Assumption: The input number will be a positive integer number >= 1 and <= 25000.

For e.g.

If the given number is 292, the program should return 2 because there are only 2 unique digits '2' and '9' in this number

If the given number is 1015, the program should return 3 because there are 3 unique digits in this number, '1', '0', and '5'.

**For example:**

| **Input** | **Result** |
| --- | --- |
| 292 | 2 |
| 1015 | 3 |

**Answer:**

﻿def digit(N):

return len(set(str(N)))

N=int(input()) print(digit(N))

|  | **Input** | **Expected** | **Got** |  |
| --- | --- | --- | --- | --- |
|  | 292 | 2 | 2 |  |
|  | 1015 | 3 | 3 |  |
|  | 123 | 3 | 3 |  |

**Ex. No. : 4.2 Date: 02/05/2024**

**Register No.: 231401013 Name: ARCHITHA. M**

**Next Perfect Square**

Given a number N, find the next perfect square greater

than N.

**Input Format:**

Integer input from stdin.

**Output Format:**

Perfect square greater than N.

**Example Input:**

10

**Output:**

16

PROGRAM:

from math import sqrt 2

n=int(input())

while int(sqrt(n))!=sqrt(n):

n=n+1

print(n)

|  | **Input** | **Expected** | **Got** |  |
| --- | --- | --- | --- | --- |
|  | 10 | 16 | 16 |  |

**Ex. No. : 4.3 Date: 02/05/2024**

**Register No.: 231401013 Name: ARCHITHA. M**

**Product of Single Digit Numbers**

Given a positive integer N, check whether it can be represented as a product of single digit numbers.

**Input Format:**

Single Integer input.

**Output Format:**

Output displays Yes if condition satisfies else prints No.

**Example Input:**

14

**Output:**

Yes

**Example Input:**

13

**Output:**

No

**PROGRAM:**

a=int(input())

if a%2==0 or a%3==0 or a%5==0 or a%7==0 or a%9==0:

print("Yes")

else:

print("No")

|  | **Input** | **Expected** | **Got** |  |
| --- | --- | --- | --- | --- |
|  | 14 | Yes | Yes |  |
|  | 13 | No | No |  |

**Ex. No. : 4.4 Date: 02/05/2024**

**Register No.: 231401013 Name: ARCHITHA. M**

**Fibonacci series**

Write a program to return the nth number in the fibonacci series.

The value of N will be passed to the program as input.

**NOTE:** Fibonacci series looks like -

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, ... and so on.

i.e. Fibonacci series starts with 0 and 1, and continues generating the next number

as the sum of the previous two numbers.

• first Fibonacci number is 0,

• second Fibonacci number is 1,

• third Fibonacci number is 1,

• fourth Fibonacci number is 2,

• fifth Fibonacci number is 3,

• sixth Fibonacci number is 5,

• seventh Fibonacci number is 8, and so on.

**For example:**

| **Input** | **Result** |
| --- | --- |
| 1 | 0 |
| 4 | 2 |

**PROGRAM:**

n=int(input())

if n<2:

print(n-1)

else:

n=n-1

fs=[0,1]

for i in range(1,n):

fs.append(fs[i]+fs[i-1])

print(fs[n])

|  | **Input** | **Expected** | **Got** |  |
| --- | --- | --- | --- | --- |
|  | 1 | 0 | 0 |  |
|  | 4 | 2 | 2 |  |
|  | 7 | 8 | 8 |  |

**Ex. No. : 4.5 Date: 02/05/2024**

**Register No.: 231401013 Name: ARCHITHA. M**

**Prime or Not**

Write a program that finds whether the given number N is Prime or not.

If the number is prime, the program should return 2 else it must return 1.

Assumption: 2 <= N <=5000, where N is the given number.

Example1: if the given number N is 7, the method must return 2

Example2: if the given number N is 10, the method must return 1

**For example:**

| **Input** | **Result** |
| --- | --- |
| 7 | 2 |
| 10 | 1 |

**PROGRAM:**

n=int(input())

f=0

for i in range(2,n):

if n%i==0:

f=1

break

if f==1:

print(1)

else:

print(2)

|  | **Input** | **Expected** | **Got** |  |
| --- | --- | --- | --- | --- |
|  | 7 | 2 | 2 |  |
|  | 10 | 1 | 1 |  |

**Ex. No. : 4.6 Date: 02/05/2024**

**Register No.: 231401013 Name: ARCHITHA. M**

**Count of Non-Repeated Digits**

Write a program to find the count of non-repeated digits in a given number N. The number will be passed to the program as an input of type int.

**Assumption:** The input number will be a positive integer number >= 1 and <= 25000.

Some examples are as below.

If the given number is 292, the program should return 1 because there is only 1 non-repeated digit '9' in this number

If the given number is 1015, the program should return 2 because there are 2 non-repeated digits in this number, '0', and '5'.

If the given number is 108, the program should return 3 because there are 3 non-repeated digits in this number, '1', '0', and '8'.

If the given number is 22, the function should return 0 because there are NO non-repeated digits in this number.

**For example:**

| **Input** | **Result** |
| --- | --- |
| 292 | 1 |
| 1015 | 2 |
| 108 | 3 |
| 22 | 0 |

**PROGRAM:**

def digits (n):

count=0

for digit in range(10):

if str(n).count(str(digit))==1:

count+=1

return count

n=int(input())

print(digits(n))

|  | **Input** | **Expected** | **Got** |  |
| --- | --- | --- | --- | --- |
|  | 292 | 1 | 1 |  |
|  | 1015 | 2 | 2 |  |
|  | 108 | 3 | 3 |  |
|  | 22 | 0 | 0 |  |

**Ex. No. : 4.7 Date: 02/05/2024**

**Register No.: 231401013 Name: ARCHITHA. M**

**Sum of the Series**

Write a program to find the sum of the series 1 +11 + 111 + 1111 + ... + n terms (n will be given as input from the user and sum will be the output)

**Sample Test Cases:**

**Test Case 1**

**Input**

4

**Output**

1234

**Test Case 2**

**Input**

6

**Output**

123456

**PROGRAM:**

import math

a=int(input())

b=(pow(10,a+1)-10-9\*a)/81

print('{:.0f}'.format(b))

|  | **Input** | **Expected** | **Got** |  |
| --- | --- | --- | --- | --- |
|  | 4 | 1234 | 1234 |  |
|  | 6 | 123456 | 123456 |  |

**Ex. No. : 4.8 Date: 02/05/2024**

**Register No.: 231401013 Name: ARCHITHA. M**

**Perfect Square after Adding 1**

Given an integer N, check whether N the given number can be made a perfect

square after adding 1 to it.

**Input Format:**

Single integer input.

**Output Format:**

Yes or No.

**Example Input:**

24 Output:

Yes

**Example Input:**

26

**Output:**

No

**For example:**

| **Input** | **Result** |
| --- | --- |
| 24 | Yes |

**PROGRAM:**

def square(n):

if n%4==0:

return "Yes"

else:

return "No"

n=int(input())

print(square(n))

|  | **Input** | **Expected** | **Got** |  |
| --- | --- | --- | --- | --- |
|  | 24 | Yes | Yes |  |
|  | 26 | No | No |  |

**Ex. No. : 4.9 Date: 02/05/2024**

**Register No.: 231401013 Name: ARCHITHA. M**

**Disarium Number**

A Number is said to be Disarium number when the sum of its digit raised to the power of their respective positions becomes equal to the number itself. Write a program to print number is Disarium or not.

**Input Format:**

Single Integer Input from stdin.

**Output Format:**

Yes or No.

**Example Input:**

175

**Output:**

Yes

**Explanation**

1^1 + 7^2 +5^3 = 175

**Example Input:**

123

**Output:**

No

**For example:**

| **Input** | **Result** |
| --- | --- |
| 175 | Yes |
| 123 | No |

**PROGRAM:**

def dis (num):

num\_str=str(num)

sum=0

for i in range(len(num\_str)):

sum+=int(num\_str[i])\*\*(i+1)

if sum==num:

return "Yes"

else:

return "No"

num=int(input())

print(dis (num))

|  | **Input** | **Expected** | **Got** |  |
| --- | --- | --- | --- | --- |
|  | 175 | Yes | Yes |  |
|  | 123 | No | No |  |

**Ex. No. : 4.10 Date: 02/05/2024**

**Register No.: 231401013 Name: ARCHITHA. M**

**Factors of a Number**

Determine the factors of a number (i.e., all positive integer values that evenly divide into a number.

**For example:**

| **Input** | **Result** |
| --- | --- |
| 20 | 1 2 4 5 10 20 |

**PROGRAM:**

n=int(input())

for i in range(1,n+1):

if(n%i==0):

print(i,end=' ')

else:

continue

|  | **Input** | **Expected** | **Got** |  |
| --- | --- | --- | --- | --- |
|  | 20 | 1 2 4 5 10 20 | 1 2 4 5 10 20 |  |
|  | 5 | 1 5 | 1 5 |  |
|  | 13 | 1 13 | 1 13 |  |