# 04 - Iteration Control Structures

**Ex. No. : 4.1** **Date:**

**Register No.:** **Name:**

---

# Factors of a number

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

```
n = int(input())
fact=[]
for i in range(1,n+1):
if n%i == 0:
fact.append(i)
for factor in fact:
print(factor, end=" ")
```

**Ex. No. : 4.2** **Date:**

**Register No.:** **Name:**

---

# Non Repeated Digit Count

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.

```
a=int(input())
1=[]
c=str(a)
k=len(str(a))
count=0
n=0
for i in range (0,k):
1.append(c[i])
for i in range (0,k):
flag=0
for j in range (0,k):
if(1[i] == 1[j]):
flag+=1
if(flag == 1):
count+=1

        print(count)
```

**Ex. No. : 4.3** **Date:**

**Register No.:** **Name:**

---

# Prime Checking

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.

```
h = int(input())
count=0
if(n>=2 and n <= 5000):
for i in range(2,n):
if(n%i == 0):
print(1)
count+=1
break
if(count == 0):
print(2)
```

**Ex. No. : 4.4** **Date:**

**Register No.:** **Name:**

---

# Next Perfect Square

Given a number N, find the next perfect square greater than N.

```
n = int(input())
for i in range(1,10):
p=i*i
if(p>n):
break

  print(p)
```

**Ex. No. : 4.5** **Date:**

**Register No.:** **Name:**

---

# Nth Fibonacci

Write a program to return the nth number in the fibonacci series. The value of N will be passed to the program as input.

```
n=int(input())
temp= 0
a = 1
s = 0
for i in range (0, n) :
s =temp+a
a=temp
temp= s

        print(a)
```

**Ex. No. : 4.6** **Date:**

**Register No.:** **Name:**

---

# 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.

```
n =int(input())
k=len(str(n))
sum = 0
c=n
count =k
for i in range(0,k):
temp=c%10
sum+=temp ** count
count -= 1
c//=10
if(sum == n):
print("Yes")
else:
print("No")
```

**Ex. No. : 4.7** **Date:**

**Register No.:** **Name:**

---

## Sum of 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)

```
n = int(input())
m = 0
V=1
for i in range(1,n+1):
m+=V
V=(v*10)+1
print(m)
```

**Ex. No. : 4.8** **Date:**

**Register No.:** **Name:**

---

# Unique Digit Count

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'.

```
n=int(input())
k=str(n)
g=len(str(n))
count=0
v =[]
if(n>=1 and n <= 25000):
for i in range(0,g):
if(k[i] not in v ):
count+=1
v.append(k[i])

    print(count)
```

**Ex. No. : 4.9** **Date:**

**Register No.:** **Name:**

---

# Product of single digit

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

```
n = int(input())
while n%2 == 0:
n//=2
while n%3 == 0:
n//=3
while n%5 == 0:
n//=5
while n%7 == 0:
n//=7
if(n == 1):
print("Yes")
else:

        print("No")
```

**Ex. No. : 4.10** **Date:**

**Register No.:** **Name:**

---

# Perfect Square After adding One

Given an integer N, check whether N the given number can be made a perfect square after adding 1 to it.

```
n = int(input())
m=n+1
if int(m ** 0.5) ** 2 == m:
print("Yes")
else:
print("No")|
```