Date
6-8-25

TASK-2 : Implement Conditional, Control and looping statements

2.1. Factorial of a number

AIM : Calculate factorial of a number using while loop.

Algorithm

step 1 : start
step 2 : Input the number n
step 3 : if n<0, print "Error: Negative number" and stop
step 4 : initialize: i=1 & fact=1
step 5 : while i≤n, repeat steps 6-7
step 6 : fact = fact x i
step 7 : i = i+1
step 8 : print the value of fact
step 9 : stop

~~Outpu~~ INPUT :

Enter a number to calculate its factorial :

OUTPUT :

The factorial of 17 is 355687428096000.

RESULT : Thus, the calculating factorial of a number using while loop was Executed succesfully.

Program

```
#Factorial using while loop
#input from user
num = int(input("Enter a number to calculate its factorial:"))
#check if the number is negative
if num<0:
    Print("Factorial does not exist for negative numbers.")
elif num==0:
    Print("The factorial of 0 is 1.")
else:
    factorial = 1
    i=1:
    while i<=num:
    factorial *= i
    i+=1
    Print(f"The factorial of {num} is {factorial}.")
```

2.2 Counting the non repeated digits in a given number

AIM: To 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.

Algorithm:

step 1: Start
step 2: Input the Number N
step 3: Initialize a hash table (or array) of size 10, set all values to 0
step 4: for each digit in N: increment the count of that digit in the hash table
step 5: Initialize a variable, Count = 0
step 6: output the values of Count (no. of non-repeated digits)
step 8: stop

INPUT:

Enter a number (1 to 25000): 17

OUTPUT:

Number of non-repeated digits: 2

PROGRAM:

```
#input from user
num = int(input("Enter a number (1 to 25000): "))
#check if input is within the valid range
if num<1 or num > 25000:
    Print("The number must be between 1 and 25000.")
else:
    digit_count = {}
    for digit in str(num):
        if digit in digit_count:
            digit_count[digit] += 1
        else:
            digit_count[digit] = 1
    #count digits that occured only once
    non_repeated_count = sum(1 for count in digit_count.
        values() if count == 1)
    Print(f"Number of non-repeated digits:
    {non_repeated_count}")
```

2.3 Multiplication Table Generator

AIM: To print multiplication table of a number up to 10.

Algorithm:

Step 1: Start
Step 2: Input number n
Step 3: Initialize a Counter variable i to 1
Step 4: a: product result = n * i
b. print result in format "n x i = result"
c. increment i by 1
Step 5: stop

Input:

Enter a number : 7
multiplication table of 7

Output:

7 * 1 = 7
7 * 2 = 14
7 * 3 = 21
7 * 4 = 28
7 * 5 = 35
7 * 6 = 42
7 * 7 = 49
7 * 8 = 56
7 * 9 = 63
7 * 10 = 70

Result: Thus, the printing multiplication table of a number up to 10 has been Executed successfully.

Programming:

```
# multiplication Table Generator
# input from user
num = int (input ("Enter a number: "))
# print multiplication table up to 10
Print (f"multiplication table of {num}:")
i = 1
while i <= 10:
    print (f"{num} x {i} = {num * i}")
    i += 1
```

2.4 Automorphic number

AIM: An automorphic number is a number whose square ends with the number itself

Algorithm:

step 1: start
step 2: input the number n
step 3: calculate the square of n and store it in square
step 4: count the number of digits in n (let's call it count)
step 5: Compute last digits = square % (10^count) - this extracts the last count digits of the square
step 6: if last digits is equal to n, then
- print "Automorphic
Else
- print "Not Automorphic"

step 7: stop

Input:-

Enter a number: 23

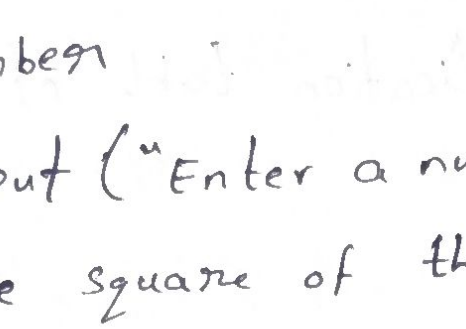
output:

Not Automorphic

RESULT: Thus, the an automorphic number is a number whose square ends with the number itself was verified and Executed Succesfully.

Program :

```
#input number
num = int (input ("Enter a number: "))
# calculate square of the number
square = num * num
# convert both to string to check ending digits
num_str = str (num)
square_str = str (square)
# check if square ends with the number
if square_str.endswith (num_str):
    print ("Automorphic")
else:
    print ("Not Automorphic")
```



2.5 Counting the number of prime numbers in a specified range

Aim: To write a program to find the count of the number of prime numbers in a specified range

Algorithm:

Step 1: start
Step 2: input the starting number start and ending number end
Step 3: initialize count to 0 (to store the number of primes)
Step 4: for each number num from start to end (inclusive):
    a. if num less than 2, skip it
    b. check if num is prime
        i. For each integer i from 2 to √num:
step 5. print the values of count
step 6. stop.

Input:

Enter starting number of rang (>=2): 7
Enter Ending number of range (<=7919): 99

Output:

Number of primes between 7 and 99: 22

Result: Thus, the Program to find the Count of the number of prime numbers in a specified range has been executed succesfully.

VEL TECH	
EX No.	
PERFORMANCE (5)	
RESULT AND ANALYSIS (5)	
VIVA VOCE (5)	
RECORD (5)	
TOTAL (20)	
SIGN WITH DATE	

Program

```
start_range = int(input("Enter the starting number of
                the range:"))
end_range = int(input("Enter the ending number of the
                range:"))
    if start_range > end_range:
        Print("Error: The starting number Cannot be
              greater than the Ending number.")
    Else:
        Prime_Count = 0
        for num in range (start_range_End_range+1):
        if num <= 1
            Countine.
        is_prime = True
        for in range (2, int (num ** 0.5)+1):
            if num % i == 0:
            is_prime = false
            break
        if is_prime

Expect value error:
    Print ("Invalid input. please Enter integers
           for the range")
```

RESULT: Thus the program to find Count of numbers of prime number in a specified range has been Executed successfully.

6/8

VELTECH	
EX No.	2
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	5
RECORD (5)	
TOTAL (20)	20
SIGN WITH DATE	