1. Factorial

Aim

To print the factorial of a given number using a while loop.

Algorithm

```
Input: A number
Output: The factorial of the number

Step 1: Start
Step 2: Read a number, say, num.
Step 3: Read a loop variable, say, i and initialise i=1.
Step 4: Initialise factorial=1.
Step 5: Check if i<=num. If not, exit the loop and go to Step 8.
Step 6: If yes, multiply the factorial by i and store it as the factorial.
Step 7: Add 1 to i. Go to Step 5.
Step 8: End
```

Program

```
# Python program to print the factorial of a given number

num=int(input("Enter a number: "))
fact,i=1,1
while i<=num:
    fact*=i
    i+=1
print("The factorial is: ",fact)
```

Output

```
Input:
6

Output:

Enter a number: 6
The factorial is:  720
```

Results / Inferences

Program for printing the factorial of a given number using a while loop is written and executed.

2. Armstrong number

Aim

To check if a given number is an Armstrong number or not using a while loop.

Algorithm

Input: A number
Output: Conclusion of whether the number is an Armstrong number or not

Step 1: Start
Step 2: Read a number, say, num.
Step 3: Store num in a temporary variable, i.e., temp.
Step 4: Initialise sum=0.
Step 5: Check if temp=0. If yes, exit the loop and go to Step 8.
Step 6: If not, add the cube of the one’s digit of temp to sum.
Step 7: Divide temp by 10 (through floor division) and go to Step 5.
Step 8: Check if sum equals num. If yes, print that it is an Armstrong number. If no, print that it is not an Armstrong number.
Step 9: End

Program

```
# Python program to check if a given number is an Armstrong number

num=int(input("Enter a number: "))
temp=num
sum=0
```

```
while (temp!=0):
    sum+=(temp%10)**3
    temp//=10
if sum==num:
    print("Armstrong number")
else:
    print("Not an Armstrong number")
```

Output

Input:
153, 121

Output:

```
Enter a number: 153
Armstrong number

Enter a number: 121
Not an Armstrong number
```

Results / Inferences

Program for printing the factorial of a given number using a while loop is written and executed.

3. Greatest Common Divisor

Aim

To find the GCD of two numbers using the Euclidean algorithm.

Algorithm

Input: Any two numbers
Output: The GCD of the numbers

Step 1: Start
Step 2: Read two numbers, say, a and b.
Step 3: If a is less than b, interchange a and b. Hence, a is greater than b.
Step 4: If b is 0, store the GCD as a and go to Step 7.
Step 5: Take the remainder of dividing a by b and check if it is 0. If yes, store the GCD as b and go to Step 7.
Step 6: If no, assign the value of b to a and that of r to b. Go to Step 5.
Step 7: Print the GCD.
Step 8: End

Program

```
# Python program to find the GCD of two numbers

num1=int(input("Enter first number: "))
num2=int(input("Enter second number: "))
a=max(num1,num2)
b=min(num1,num2)
if (a==0 and b==0):
    print("GCD is 0.")
elif (b==0):
    print("GCD: ",a)
else:
    r=a%b
    while (r!=0):
        a=b
        b=r
        r=a%b
    print("GCD: ",b)
```

Output

```
Input:
45 60

Output:

Enter first number: 45
Enter second number: 60
GCD:  15
```

Results / Inferences

Program for finding the GCD of a given number using the Euclidean algorithm is written and executed.

4. Fibonacci series

Aim

To print the Fibonacci series for a given number of terms using a for loop.

Algorithm

Input: The number of terms
Output: The Fibonacci series till that term

```
Step 1: Start
Step 2: Read the number of terms, say, n.
Step 3: Read a loop variable, say, i and initialise i=0.
Step 4: Initialise two variables as a=0 and b=1.
Step 5: Check if i<n-2. If not, exit the loop and go to Step 8.
Step 6: If yes, assign the value of the next term as the sum of the previous two terms and print the term.
Step 7: Add 1 to i. Go to Step 5.
Step 8: End
```

Program

```
# Python program to print the Fibonacci series

n=int(input("Enter number of numbers in series: "))
a,b=0,1
print(a,b, end=" ")
for i in range (0,n-2):
  c=a+b
  a,b=b,c
  print(c, end=" ")
```

Output

```
Input:
8

Output:

Enter number of numbers in series: 8
0 1 1 2 3 5 8 13
```

Results / Inferences

Program for printing the Fibonacci series for a given number of terms using the for loop is written and executed.

5. Multiplication table

Aim

To print the multiplication table of a given number using a for loop.

Algorithm

```
Input: A number
Output: The multiplication table till 10 of the number

Step 1: Start
Step 2: Read a number.
Step 3: Read a loop variable, say, i and initialise i=1.
Step 4: Check if i<=10. If not, exit the loop and go to Step 8.
Step 5: If yes, multiply the number by i and store it as the product.
Step 6: Print the multiplication for this iteration.
Step 7: Add 1 to i. Go to Step 4.
Step 8: End
```

Program

```python
# Python program to print the multiplication table of a number

num=int(input("Enter a number: "))
for i in range (1, 11):
  print(num,"*",i,"=",num*i)
```

Output

```
Input:
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
```

Results / Inferences

Program for printing the multiplication table of a given number using a for loop is written and executed.

UGE1197
Programming in Python Lab
AY: 2020-2021

Krithika Swaminathan
Date:
S03018

Ex. No.: 4

6. Patterns

Aim

To print the given patterns.

Algorithm

a. Pattern 1:

Input: NIL
Output: The required pattern

```
Step 1: Start
Step 2: Read two loop variables, say, i and j.
Step 3: Initialise i=1 and j=1.
Step 4: Check if i<6. If not, exit the loop and go to Step 7.
Step 5: If yes, check if j<i+1. If not, add 1 to i and go to Step 4.
Step 6: If yes, print an asterisk. Add 1 to j and go to Step 5.
Step 7: Initialise i=4.
Step 8: Check if i>0. If not, exit the loop and go to Step 12.
Step 9: If yes, initialise j=i.
Step 10: Check if j>0. If not, subtract 1 from i and go to Step 8.
Step 11: If yes, print an asterisk. Subtract 1 from j and go to Step 10.
Step 12: End
```

b. Pattern 2:

Input: NIL
Output: The required pattern

UGE1197
Programming in Python Lab
AY: 2020-2021

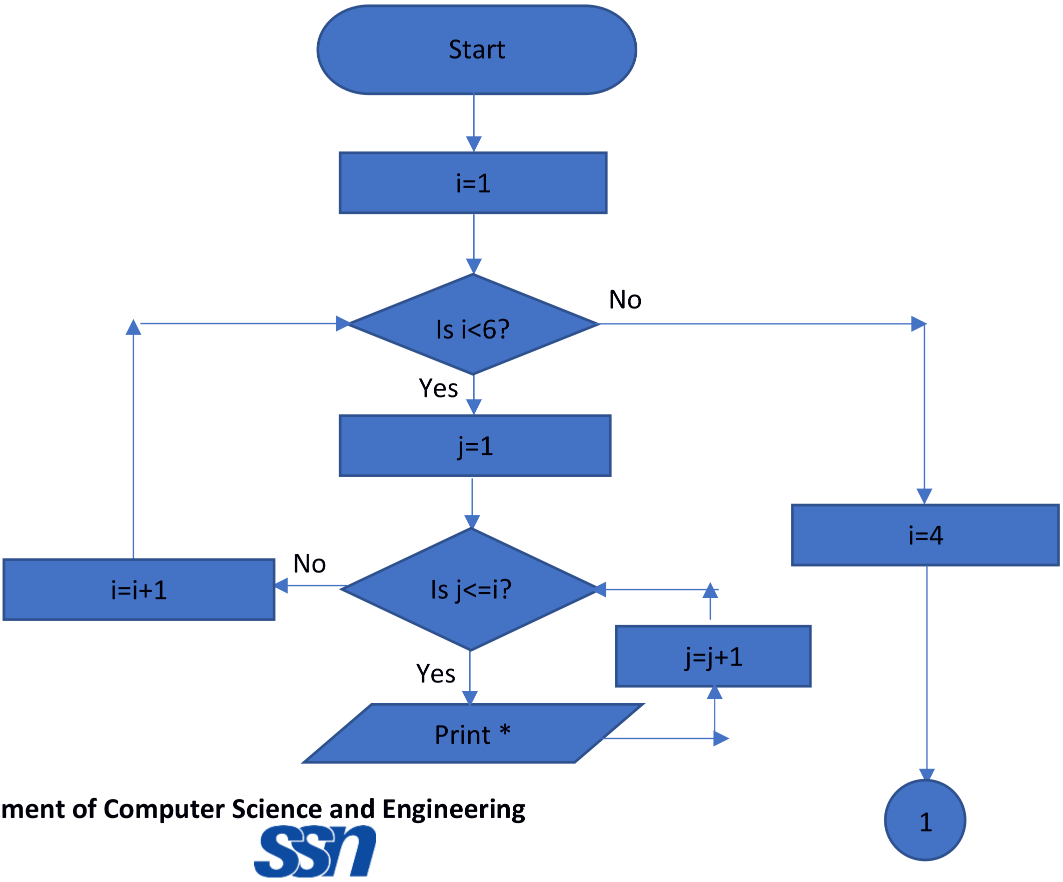
Krithika Swaminathan
Date:
S03018

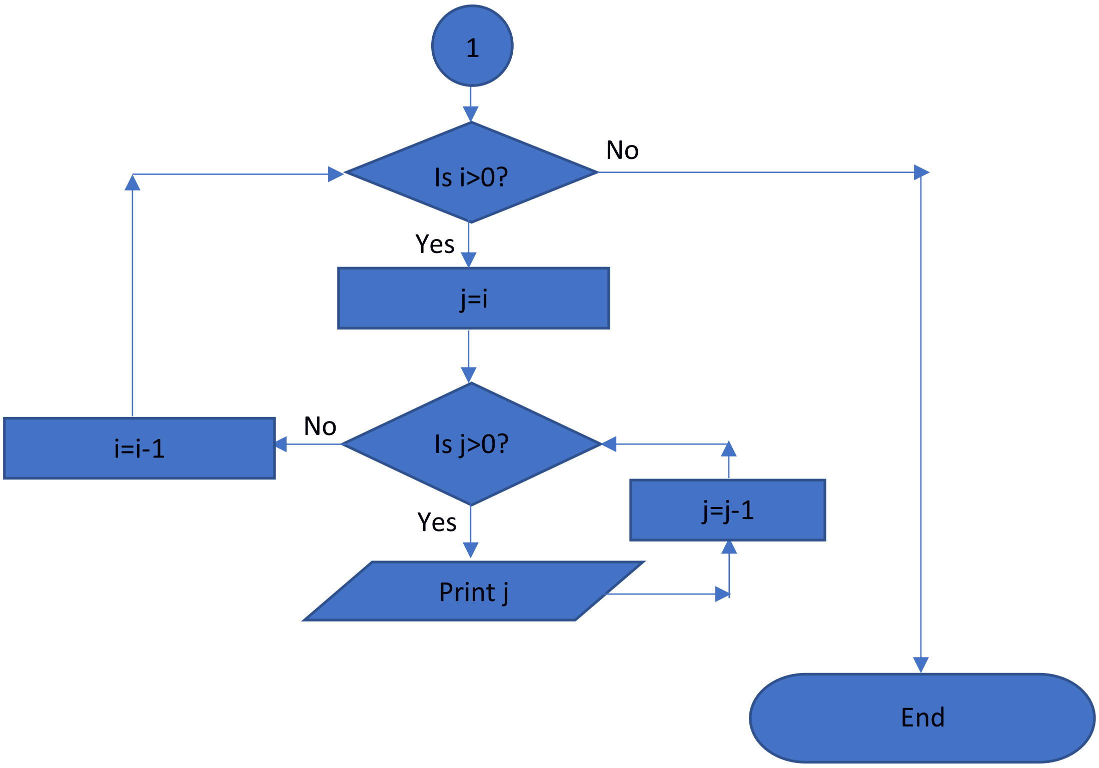
Ex. No.: 4

```
Step 1: Start
Step 2: Read two loop variables, say, i and j.
Step 3: Initialise i=1.
Step 4: Check if i<6. If not, exit the loop and go to Step 8.
Step 5: If yes, initialise j=i.
Step 6: Check if j>0. If not, add 1 to i and go to Step 4.
Step 7: If yes, print the value of j. Subtract 1 from j and go
to Step 6.
Step 8: End
```

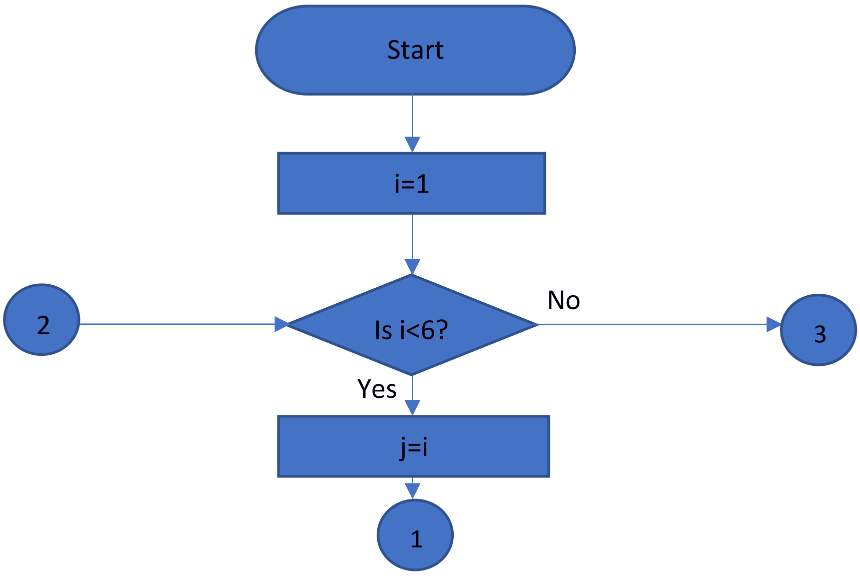
Flowchart

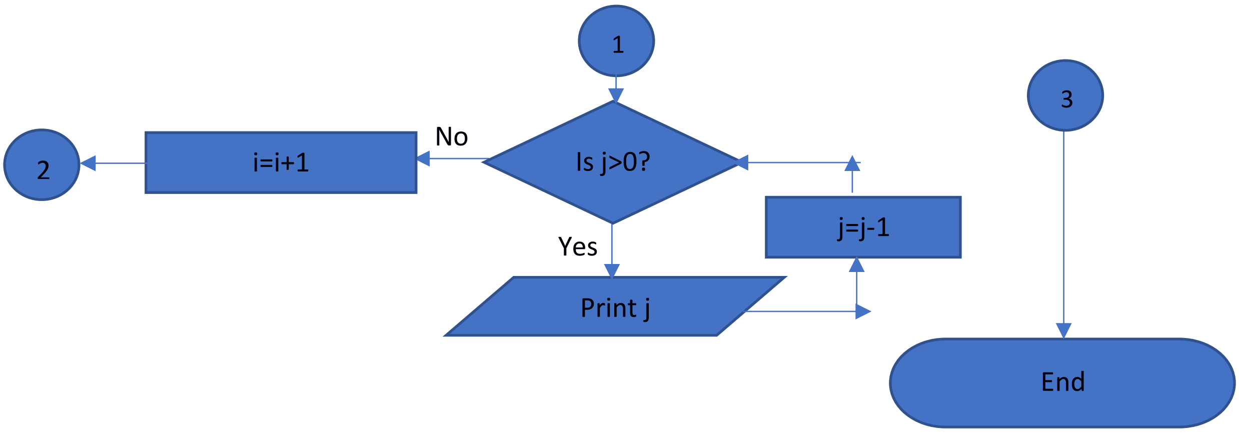
a. Pattern 1:




1
Is i>0?
No
Yes
j=i
Is j>0?
No
i=i-1
Yes
j=j-1
Print j
End

b. Pattern 2:


Start
i=1
2
Is i<6?
No
3
Yes
j=i
1



Program

```
# Python program to print the given patterns

#pattern 1
for i in range (1,6):
  for j in range (1,i+1):
    print("*",end=" ")
  print()
for i in range (4,0,-1):
  for j in range (i,0,-1):
    print("*",end=" ")
  print()
print()

#pattern 2
for i in range (1,6):
  for j in range (i,0,-1):
    print(j,end=" ")
  print()
print()
```

Output

```
*
* *
* * *
* * * *
* * * * *
* * * *
* * *
* *
*

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

Results / Inferences

Program for printing the required patterns is written and executed.

7. Series

Aim

To print the sum of the given series.

Algorithm

a. Series A:

```
Input: The number of terms
Output: The sum of the series A.

Step 1: Start
Step 2: Read the number of terms, say, n.
Step 3: Read a loop variable, say, i.
Step 4: Initialise i=1 and sum=0.
Step 5: Check if i<=n. If not, exit the loop and go to Step 8.
Step 6: If yes, add the reciprocal of i to sum.
Step 7: Add 1 to i and go to Step 5.
Step 8: Print the sum of the series in the given format.
Step 9: End
```

b. Series B:

```
Input: The number of terms and the value of x
Output: The sum of the series B.

Step 1: Start
Step 2: Read the number of terms and value of x, say, n and x.
Step 3: Read a loop variable, say, i.
Step 4: Initialise i=2 and sum=1.
Step 5: Check if i<=n. If not, exit the loop and go to Step 8.
```

Step 6: If yes, calculate the i^{th} power of x and divide it by i. Then, add the result to sum.
Step 7: Add 1 to i and go to Step 5.
Step 8: Print the sum of the series in the given format.
Step 9: End

Program

```
# Python program to print the sum of the given series

# Series A
print("Series A")
n=int(input("Enter number of terms: "))
sum=0
for i in range (1,n+1):
  sum+=(1/i)
  if i==n:
    print("1/″,i,"=",sum)
  else:
    print("1/",i,"+ ",end="")
print()
# Series B
print("Series B")
n=int(input("Enter number of terms: "))
x=float(input("Enter value of x: "))
sum=1
print("1",end="")
for i in range (2,n+1):
  sum+=(x**i)/i
  print("+ x^",i,"/",i,end="")
print(" =",sum)
```

Output

```
Input:

3
4 2

Output:

Series A
Enter number of terms: 3
1/1 + 1/2 + 1/3 = 1.8333333333333333

Series B
Enter number of terms: 4
Enter value of x: 2
1 + x^2/2 + x^3/3 + x^4/4 = 9.666666666666666
```

Results / Inferences

Program for printing the sum of the given series using a for loop is written and executed.

8. Integers 0-7 without 3 and 6

Aim

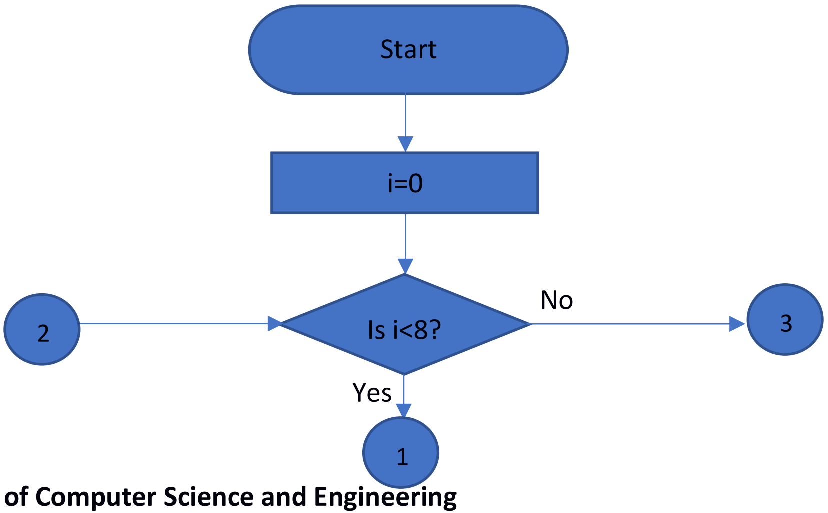
To print all integers from 0 to 7 except 3 and 6 using the continue statement.

Algorithm

Input: NIL
Output: Integers from 0 to 7 except 3 and 6

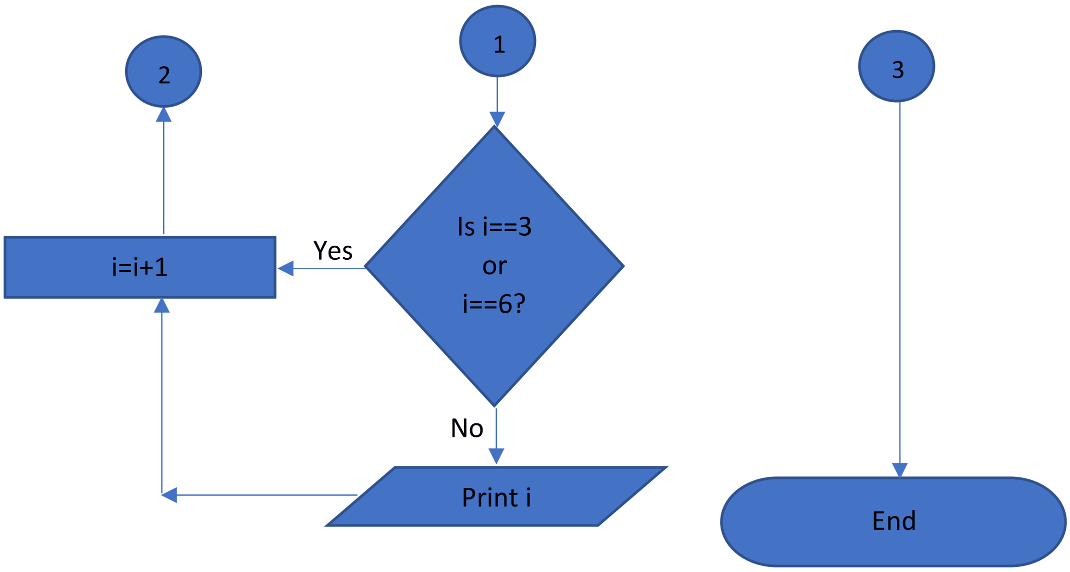
```
Step 1: Start
Step 2: Read a loop variable, say, i and initialise i=0.
Step 3: Check if i<=7. If not, exit the loop and go to Step 7.
Step 4: If yes, check if the value of i equals 3 or 6.
Step 5: If yes, go to Step 3. If no, go to Step 6.
Step 6: Print the value of i and add 1 to i. Go to Step 3.
Step 7: End
```

Flowchart



Krithika Swaminathan
Date:
S03018

Ex. No.: 4



Program

```
# Python program to print all integers from 0 to 7 except 3 and 6
using the continue statement

for i in range(0,8):
  if (i==3 or i==6):
    continue #Using the continue statement for the next iteration
  else:
    print(i)
```

Output

```
0
1
2
4
5
7
```

Results / Inferences

Program for printing all the integers from 0 to 7 except 3 and 6 is written and executed.

9. Prime or Composite?

Aim

To classify a given number as prime or composite.

Algorithm

Input: A number
Output: Classification of prime or composite

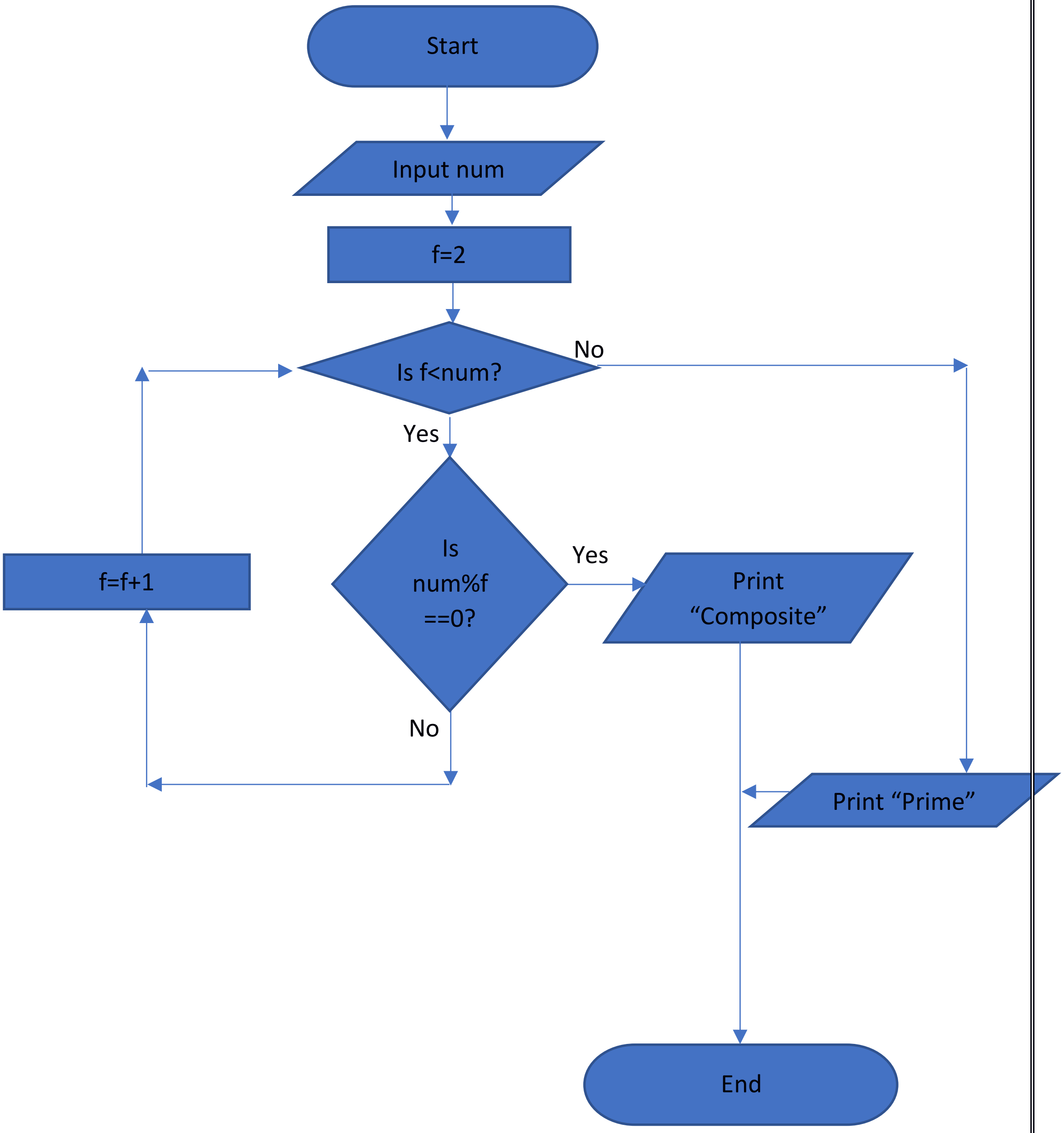
Pre-condition: The number must be a natural number
Post-condition: The number should be classified as prime, composite or neither prime nor composite.

```
Step 1: Start
Step 2: Read the number input by the user.
Step 3: Check if the number equals 1. If no, go to Step 5.
Step 4: If yes, print "Neither prime nor composite". Go to Step 10.
Step 5: Read a loop variable, say, f and initialise f=2.
Step 6: Check if f is less than the number. If no, go to Step 9.
Step 7: If yes, check if the number is divisible by f.
Step 8: If yes, print "Prime" and go to Step 9. If no, add 1 to f and go to Step 6.
Step 9: If f+1 is equal to the number, print "Composite" and go to Step 10.
Step 10: End
```

Krithika Swaminathan
Date:
S03018

Ex. No.: 4

Flowchart



Ex. No.: 4

Program

```
# Python program to classify a number as prime or composite using the
break statement in the program

num=int(input("Enter a number: "))
if num==1:
  print("Neither prime nor composite")
  exit()
for f in range(2,num):
  if (num%f==0):
    print("Composite")
    break #Using the break statement to exit if a factor is found
if f+1==num:
  print("Prime")
```

Output

Input:
4,7,1

Output:

```
Enter a number: 4
Composite

Enter a number: 7
Prime

Enter a number: 1
Neither prime nor composite
```

UGE1197
Programming in Python Lab
AY: 2020-2021

Krithika Swaminathan
Date:
S03018

Ex. No.: 4

Results / Inferences

Program for checking whether a given number is prime or composite is written and executed.