UGE1197
Programming in Python Lab
AY: 2020-2021

Krithika Swaminathan
Date: 01/02/2021
S03018

Ex. No.: 6

1. Factorial

Aim

To find the factorial of a given number using recursive functions.

Algorithm

Input: Any number
Output: The factorial of the number

Pre-condition: The number must be a whole number

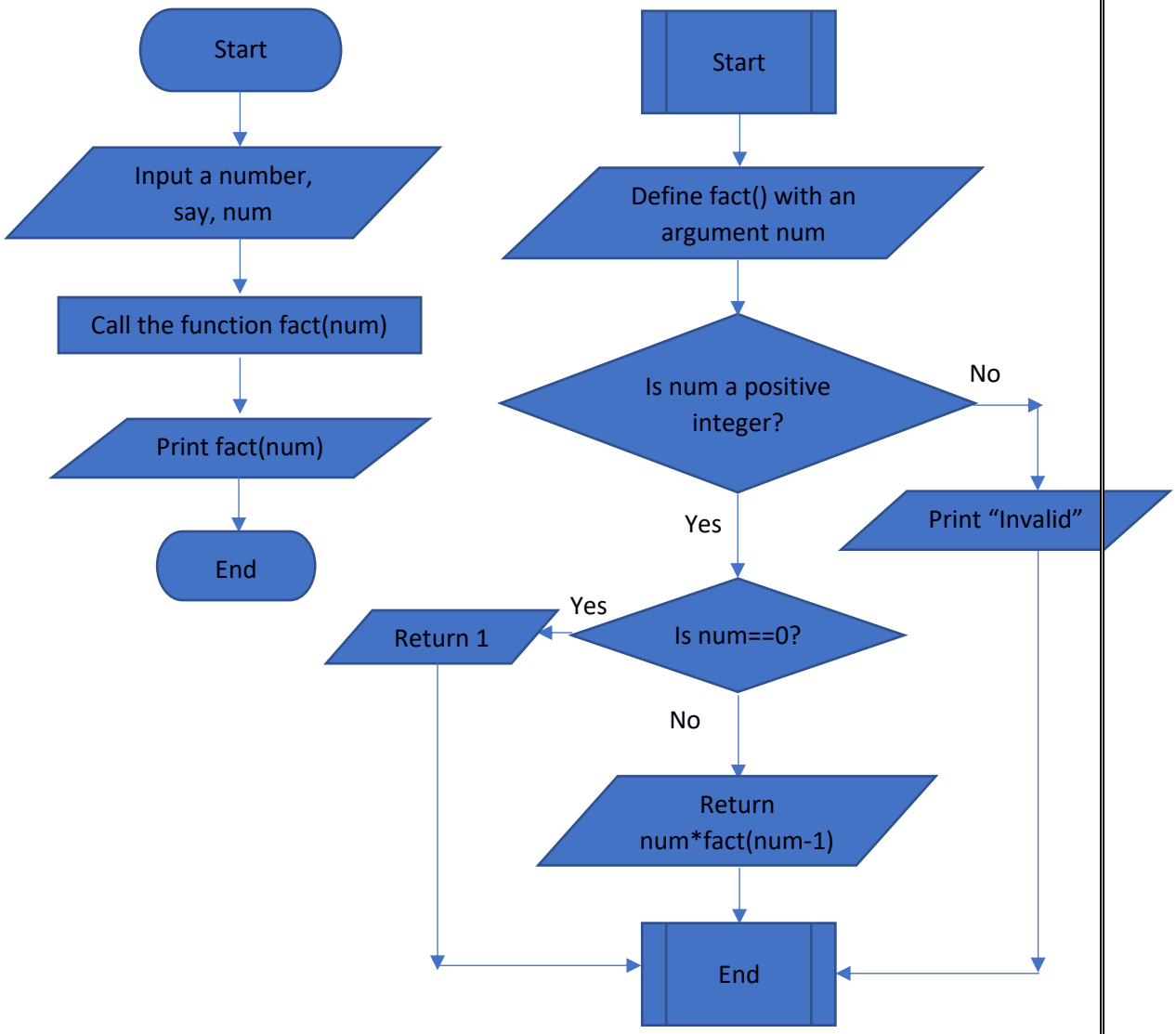
Step 1: Start
Step 2: Define a function to calculate the factorial of a number. The function accepts only whole number values.
Step 3: The function contains a base case of the argument value being 0 for which it returns the value 1. For any other value passed, the function returns the value of the number multiplied by the output of the factorial function of its predecessor.
Step 4: In the main function, the required number is obtained as input from the user and passed into the function to calculate its factorial.
Step 5: End

UGE1197
Programming in Python Lab
AY: 2020-2021

Krithika Swaminathan
Date: 01/02/2021
S03018

Ex. No.: 6

Flowchart



UGE1197
Programming in Python Lab
AY: 2020-2021

Krithika Swaminathan
Date: 01/02/2021
S03018

Ex. No.: 6

Program

```
# Python program to find and print the factorial of a given number
using recursive functions.

# Function definition
def fact(num):
    if (type(num)!=int) or (num<0):
        print("Invalid - Not a positive integer")
        return None
    if num==0:
        return 1
    else:
        return num*fact(num-1)

# to get the input from the user
n=eval(input("Enter a number: "))
#Function call
print("Factorial:",fact(n))
```

Output

Input:

4

Output:

```
Enter a number: 4
Factorial: 24
```

UGE1197
Programming in Python Lab
AY: 2020-2021

Krithika Swaminathan
Date: 01/02/2021
S03018

Ex. No.: 6

Results / Inferences

Program for finding the factorial of a given number using recursive functions is written and executed.

UGE1197
Programming in Python Lab
AY: 2020-2021

Krithika Swaminathan
Date: 01/02/2021
S03018

Ex. No.: 6

2. Fibonacci sequence

Aim

To print the Fibonacci sequence for a given number of terms using recursive functions.

Algorithm

Input: Any number of terms
Output: The corresponding term and the sequence upto that term

Pre-condition: The number must be a positive integer

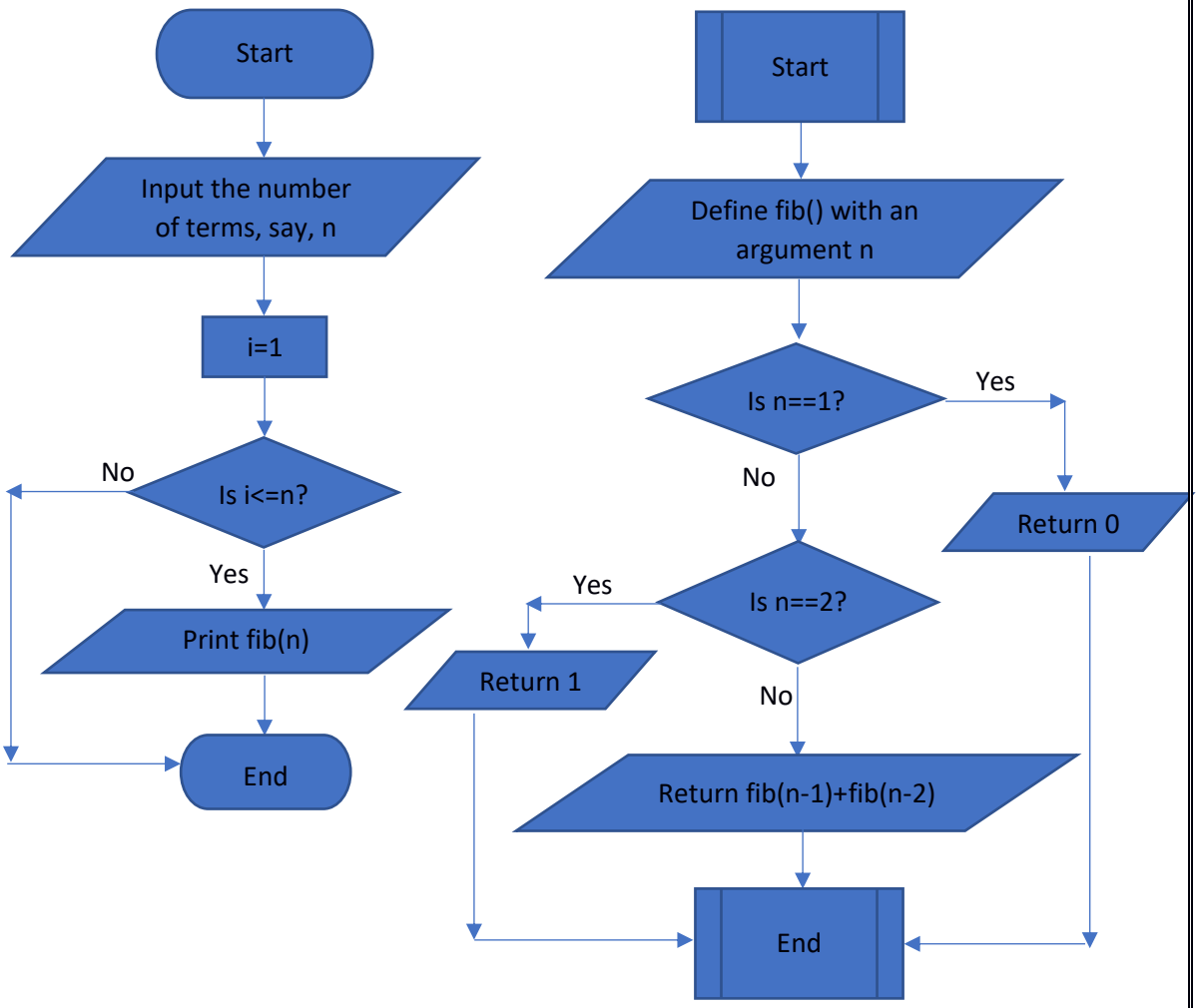
Step 1: Start
Step 2: Define a function to generate the Fibonacci sequence for a given number of terms. The function accepts the number of terms in the sequence as the argument.
Step 3: The function contains two base cases, returning 0 for the number of terms being 1 and 1 for the number of terms being 2. For any other values, the function returns the sum of the outputs of the Fibonacci function for the preceding two numbers.
Step 4: In the main function, the required number of terms is obtained as input from the user.
Step 5: Call the function passing to it every number till the required number of terms, one by one, and print them.
Step 6: End

UGE1197
Programming in Python Lab
AY: 2020-2021

Krithika Swaminathan
Date: 01/02/2021
S03018

Ex. No.: 6

Flowchart



UGE1197
Programming in Python Lab
AY: 2020-2021

Krithika Swaminathan
Date: 01/02/2021
S03018

Ex. No.: 6

Program

```
# Python program to print the Fibonacci series using recursive
functions.

# Function definition
def fib(num):
    if num==1:
        return 0
    elif num==2:
        return 1
    else:
        return fib(num-1)+fib(num-2)

n=int(input("Enter number of terms: "))
print("The term is: ",fib(n))# Function call
for i in range(1,n+1):
  print(fib(i),end=" ")# To print the series
```

Output

Input:

8

Output:

```
Enter number of terms: 8
The term is:  13
0 1 1 2 3 5 8 13
```

UGE1197
Programming in Python Lab
AY: 2020-2021

Krithika Swaminathan
Date: 01/02/2021
S03018

Ex. No.: 6

Results / Inferences

Program for printing the Fibonacci series using recursive functions is written and executed.

Krithika Swaminathan
Date: 01/02/2021
S03018

Ex. No.: 6

3. Greatest Common Divisor

Aim

To find the GCD of two numbers using recursive functions.

Algorithm

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

Pre-condition: The numbers must be positive integers

Step 1: Start
Step 2: Define a function to calculate the GCD of two numbers taking the two numbers as parameters. The function checks if the first parameter is larger than the second and if not, exchanges the two numbers.
Step 3: The function contains a base case that if the second parameter is 0, then the function should return the value of the first parameter. For any other case, the remainder of the division of the first number by the second is obtained and the function returns the output of the GCD function that contains the second number as the first parameter and the remainder as the second parameter.
Step 4: Read the two numbers as input from the user.
Step 5: Call the function and print the return value.
Step 6: End

Program

```
# Python program to find the GCD of two given numbers using recursive
functions.
```

UGE1197
Programming in Python Lab
AY: 2020-2021

Krithika Swaminathan
Date: 01/02/2021
S03018

Ex. No.: 6

```
# Function definition
def gcd(a,b):
    if a<b:
        a,b=b,a
    if b==0:
        return a
    else:
        r=a%b
        return gcd(b,r)

# to get input from the user
n1=int(input("Enter first number: "))
n2=int(input("Enter second number: "))
# Function call
print("GCD:",gcd(n1,n2))
```

Output

Input:

45 60

Output:

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

Results / Inferences

Program for finding the GCD of two numbers using recursive functions is written and executed.

Krithika Swaminathan
Date: 01/02/2021
S03018

Ex. No.: 6

4. Ackermann function

Aim

To define and execute the Ackermann function using recursive functions.

Algorithm

Input: Any two numbers, say, m and n
Output: The result of the Ackermann function

Pre-condition: The numbers should be greater than or equal to 0

Step 1: Start
Step 2: Define a function to calculate the result of the Ackermann function which takes the two numbers (m and n) as parameters. The function classifies all negative input values as invalid.
Step 3: The function contains a base case of the first parameter being 0 (m=0) for which it returns the second parameter incremented by 1, i.e., (n+1), and a case of the second parameter being 0 (n=0) for which it returns the output of the Ackermann function with the parameters- the first parameter decremented by 1 and the second parameter being 1. For any other case, the function returns the output of the Ackermann function with the first parameter equal to m-1 and the second parameter being the output of the function with parameters m and n-1.
Step 4: Read any two numbers as input from the user.
Step 5: Pass the numbers as the arguments and call the function.
Step 6: End

UGE1197
Programming in Python Lab
AY: 2020-2021

Krithika Swaminathan
Date: 01/02/2021
S03018

Ex. No.: 6

Program

```
# Python program to define and execute the Ackermann function using
recursive functions.

# Function definition
def A(m,n):
    if m==0:
        return n+1
    elif m>0 and n==0:
        return A(m-1,1)
    elif m>0 and n>0:
        return A(m-1,A(m,n-1))
    else:
        print("Invalid value")
        return None

# to get the input from the user
a=int(input("Enter the value of m: "))
b=int(input("Enter the value of n: "))
# Function call
print("The output of the Ackermann function is",A(a,b))
```

Output

Input:

1 2

Output:

```
Enter the value of m: 1
Enter the value of n: 2
The output of the Ackermann function is 4
```

UGE1197
Programming in Python Lab
AY: 2020-2021

Krithika Swaminathan
Date: 01/02/2021
S03018

Ex. No.: 6

Results / Inferences

Program for defining and executing the Ackermann function using recursive functions is written and executed.