**EXERCISE – 1**

1. Write a program to print Fibonacci series using recursion

**PROGRAM:**

def Fibonacci(n):

if n<=1:

return n

else:

return Fibonacci(n-1)+Fibonacci(n-2)

Num=int(input(“Enter the number:”))

if Num<=0:

print(“please Enter the positive number.”)

else:

print(“Fibonacci series:”)

for i in range(0,Num+1):

print(Fibonacci(i),” “,end=””)

**INPUT:**



**OUTPUT:**



**TIME COMPLEXITY:**

Time complexity of the above code is

f(n)=O(n)

2. Write a program to check the given no is Armstrong or not using recursive function.

**PROGRAM:**

def check\_armstrong(Num):

if Num==0:

return Num

else:

return pow((num%10),order)+check\_armstrong(num//10)

num=int(input("Enter a number:"))

order=len(str(num))

sum=check\_armstrong(num)

if sum==int(num):

print(Num," is an Armstrong number.")

else:

print(Num," is not an Armstrong number.")

**INPUT:**



**OUTPUT:**



**TIME COMPLEXITY:**

Time complexity of the above code is

f(n)=O(n)

3. Write a program to find the GCD of two numbers using recursive factorization

**PROGRAM:**

def gcd(a,b):

if b==0:

return a

else:

return gcd(b,a%b)

n1=int(input("Enter the number 1:"))

n2=int(input("Enter the number 2:"))

result=gcd(n1,n2)

print("The GCD of ",n1," and ",n2," is:",result)

**INPUT:**



**OUTPUT:**



**TIME COMPLEXITY:**

Time complexity of the above code is

f(n)=O(log(min(a,b))

4. Write a program to get the largest element of an array.

**PROGRAM:**

a=[10,89,9,56,4]

max=a[0]

for i in range(0,len(a)):

    if a[i]>max:

        max=a[i]

print("Largest element in the array is:",max)

**INPUT:**

a=[10,89,9,56,4]

**OUTPUT:**



**TIME COMPLEXITY:**

Time complexity of the above code is

f(n)=O(n)

5. Write a program to find the Factorial of a number using recursion.

**PROGRAM:**

def factorial(n):

    if n==0 or n==1:

        return 1

    else:

        return n\*factorial(n-1)

n=int(input("Enter the number:"))

result=factorial(n)

print("The factorial of ",n," is:",result)

**INPUT:**



**OUTPUT:**



**TIME COMPLEXITY:**

Time complexity of the above code is

f(n)=O(n)

6. Write a program for to copy one string to another using recursion

**PROGRAM:**

def copy\_string(source, destination, index=0):

    if index == len(source):

        return destination

    destination += source[index]

    return copy\_string(source, destination, index + 1)

source\_str = "Hello, World!"

destination\_str = ""

result = copy\_string(source\_str, destination\_str)

print("Source String:", source\_str)

print("Copied String:", result)

**INPUT:**

"Hello, World!"

**OUTPUT:**



**TIME COMPLEXITY:**

Time complexity of the above code is

f(n)=O(n)

7. Write a program to print the reverse of a string using recursion

**PROGRAM:**

def reverse\_string(s):

if len(s) == 0:

return s

else:

return reverse\_string(s[1:]) + s[0]

input\_string = "Hello, World!"

reversed\_string = reverse\_string(input\_string)

print("Original String:", input\_string)

print("Reversed String:", reversed\_string)

INPUT:

"Hello, World!"

**OUTPUT:**



**TIME COMPLEXITY:**

Time complexity of the above code is

f(n)=O(n)

8. Write a program to generate all the prime numbers using recursion

**PROGRAM:**

def is\_prime(n, i=2):

    if n <= 2:

        return True if n == 2 else False

    if n % i == 0:

        return False

    if i \* i > n:

        return True

    return is\_prime(n, i + 1)

print("Primes between 1 to n is:")

def generate\_primes(n):

    if n > 1:

        generate\_primes(n - 1)

        if is\_prime(n):

            print(n," ",end="")

generate\_primes(20)

**INPUT:**

Enter the number:20

**OUTPUT:**



**TIME COMPLEXITY:**

Time complexity of the above code is

f(n)=O(n)

9. Write a program to check a number is a prime number or not using recursion.

**PROGRAM:**

def is\_prime(num, i=2):

    if num <= 2:

        return num == 2

    if num % i == 0:

        return False

    if i \* i > num:

        return True

    return is\_prime(num, i + 1)

num = int(input("Enter a number: "))

if is\_prime(num):

    print(num, "is a prime number")

else:

    print(num, "is not a prime number")

**INPUT:**



**OUTPUT:**



**TIME COMPLEXITY:**

Time complexity of the above code is

f(n)=O(n)

10. Write a program for to check whether a given String is Palindrome or not using recursion

**PROGRAM:**

def is\_palindrome(s):

    s = s.lower().replace(" ", "")

    if len(s) < 2:

        return True

    if s[0] != s[-1]:

        return False

    return is\_palindrome(s[1:-1])

input\_string = "A man a plan a canal Panama"

if is\_palindrome(input\_string):

    print(f"{input\_string} is a palindrome.")

else:

    print(f"{input\_string} is not a palindrome.")

**INPUT:**

A man a plan a canal Panama

**OUTPUT:**



**TIME COMPLEXITY:**

Time complexity of the above code is

f(n)=O(n)