**Saveetha School of Engineering**

**Saveetha Institute of Medical And Technical Science**

**LAB PROGRAM - 01**

**PROGRAMMING LANGUAGE**

Python

**COURSE CODE / NAME**

CSA0666 - Design And Analysis Of Algorithm For Divide And Conquer Techniques

**SUBMITTED BY**

T Svetha – 192321155

**1. Write a program to Print Fibonacci Series using recursion**.

def fibo(n):

if n<=2:

return n

else:

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

n=int(input())

for i in range(0,n):

print(fibo(i),end=",")

**Output:**

n=5

0,1,1,2,3,5

**Time complexity:** O(n)

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

def arms(n,l):

if n==0:

return 0

else:

return(((n%10)\*\*l)+ arms(n//10,l))

n=int(input())

l=len(str(n))

if arms(n,l)==n:

print("True")

else:

print("False")

**Output:**

n=153

True

**Time complexity:** O(n)

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

def gcd(a,b):

    if a==0:

        return b

    if b==0:

        return a

    return gcd(a,b%a)

a=4,b=10

print(gcd(a,b))

**Output:**

2

**Time Complexity:** O(n)

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

lst=[5,3,6,8,4,9,2]

m=sorted(lst)

print("largest number is:",m[-1])

**Output:**

9

**Time Complexity:** O(1)

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

def fact(n):

    if n==0:

        return 1

    else:

        return n\*fact(n-1)

n=int(input())

print(fact(n))

**Output:**

n=5

120

**Time Complexity:** O(n)

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

def cs(ori):

if len(ori) == 0:

return ""

else:

return ori[0] + cs(ori [1:])

ori = "python”

copy = cs(ori)

print("Original string:", ori)

print("Copied string:", copy)

**Output:**

Original String: python

Copied String: python

**Time Complexity:** O(n)

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

def rs (s):

if len(s)==0:

return s

else:

return rs(s[1:])+s[0]

n="hi"

print("rev string:", rs(n))

**Output:**

Rev string: ih

**Time Complexity:** O(n)

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

def generate\_primes(n):

primes = [True] \* (n + 1)

primes[0] = primes[1] = False

i = 2

while i \* i <= n:

if primes[i]:

for j in range(i \* i, n + 1, i):

primes[j] = False

i += 1

return [i for i, is\_prime in enumerate(primes) if is\_prime]

primes = generate\_primes(100)

print(primes)**Output:**

**Time Complexity:** O(n^(3/2))

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

def prime(n, i=2):

if n ==i:

return True

elif n % i == 0:

return False

return prime(n, i + 1)

n=int(input())

if prime(n):

print(“prime number.")

else:

print(“ not a prime number.")

**Output:**

[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97]

**Time Complexity:** O(nlogn)

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

def is\_palindrome(s):

if len(s) < 2:

return True

if s[0] != s[-1]:

return False

return is\_palindrome(s[1:-1])

input\_string = "mom"

if is\_palindrome(input\_string):

print(“palindrome")

else:

print(“not a palindrome")

**Output:**

palindrome.

**Time Complexity:** O(n)