**LAB CYCLE – 4**

**PROGRAM NO : 01**

**Aim:**

**Write a program to print the Fibonacci series using recursion.**

**Source Code:**

**def fibonacci(n):**

**if n<=1:**

**return n**

**else:**

**return fibonacci(n-1)+fibonacci(n-2)**

**num=int(input("Enter the number of terms:"))**

**if num<0:**

**print("Please enter a positive integer")**

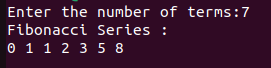
**else:**

**print("Fibonacci Series :")**

**for i in range(num):**

**print(fibonacci(i),end=" ")**

**Output:**

****

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**PROGRAM NO : 02**

**Aim:**

**Write the to implement a menu-driven calculator. Use separate functions for the different operations.**

**Source Code:**

**def addition(a,b):**

**return a+b**

**def subtraction(a,b):**

**return a-b**

**def multiplication(a,b):**

**return a\*b**

**def division(a,b):**

**return a/b**

**a=int(input("Enter the first number:"))**

**b=int(input("Enter the second number:"))**

**while True:**

**print("1.Addition\n2.Subtraction\n3.Multiplication\n4.Division\n5.Exit")**

**c=input("Enter your choice:")**

**if c=='5':**

**break;**

**elif c=='1':**

**print(f"{a} + {b} = {addition(a,b)}\n")**

**elif c=='2':**

**print(f"{a} - {b} = {subtraction(a,b)}\n")**

**elif c=='3':**

**print(f"{a} \* {b} = {multiplication(a,b)}\n")**

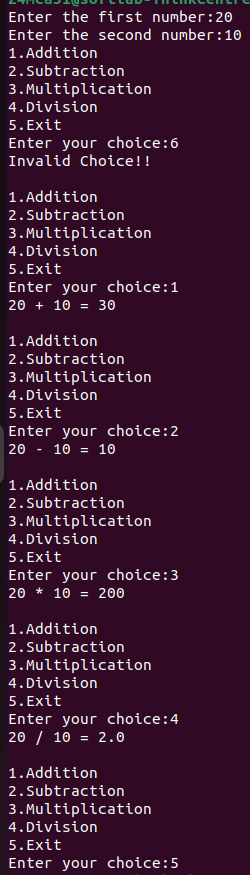
**elif c=='4':**

**print(f"{a} / {b} = {division(a,b)}\n")**

**else:**

**print("Invalid Choice!!\n")**

**Output:**

****

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**PROGRAM NO : 03**

**Aim:**

**Write a program to print the nth prime number. [Use function to check whether a number is prime or not]**

**Source Code:**

**PROGRAM NO : 04**

**Aim:**

**Write lambda functions to find the area of square, rectangle and triangle.**

**Source Code:**

**area\_square=lambda side:side\*side**

**area\_rectangle=lambda length,breadth:length\*breadth**

**area\_triangle=lambda base,height:0.5\*base\*height**

**side=float(input("Enter the side of square in centimetres:"))**

**print("Area of Square :",area\_square(side),"sq.cm")**

**length=float(input("Enter the length of rectangle in centimetres:"))**

**breadth=float(input("Enter the breadth of rectangle in centimetres:"))**

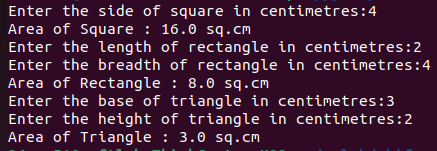
**print("Area of Rectangle :",area\_rectangle(length,breadth),"sq.cm")**

**base=float(input("Enter the base of triangle in centimetres:"))**

**height=float(input("Enter the height of triangle in centimetres:"))**

**print("Area of Triangle :",area\_triangle(base,height),"sq.cm")**

**Output:**

****

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**PROGRAM NO : 05**

**Aim:**

**Write a program to display powers of 2 using anonymous function. [ Hint use map and lambda function)**

**Source Code:**

**n=int(input("Enter the number of terms:"))**

**powers=list(map(lambda x:2\*\*x,range(n)))**

**print("Powers of 2 : ",powers)**

**Output:**

****

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**PROGRAM NO : 06**

**Aim:**

**Write a program to sum the series 1/1! + 4/2! + 27/3! + ….. + nth term. [ Hint Use a function to find the factorial of a number].**

**Source Code:**

**def factorial(num):**

**if num==0 or num==1:**

**return 1**

**else:**

**result=1**

**for i in range(2,num+1):**

**result\*=i**

**return result**

**def sum\_series(n):**

**series\_sum=0**

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

**term=(i\*\*i)/factorial(i)**

**series\_sum+=term**

**return series\_sum**

**n=int(input("Enter a positive integer:"))**

**if n<0:**

**print("Please enter a positive integer.")**

**else:**

**result=sum\_series(n)**

**print(f"The sum of the series 1/1! + 4/2! + 27/3! + .... upto {n} terms is: {result}")**

**Output:**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**PROGRAM NO : 07**

**Aim:**

**Write a function called compare which takes two strings S1 and S2 and an integer n as arguments. The function should return True if the first n characters of both the strings are the same else the function should return False.**

**Source Code:**

**def compare(s1,s2,n):**

**if s1[:n]==s2[:n]:**

**return True**

**else:**

**return False**

**s1=input("Enter first String:")**

**s2=input("Enter second String:")**

**n=int(input("Enter an integer:"))**

**result=compare(s1,s2,n)**

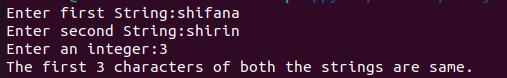
**if result==True:**

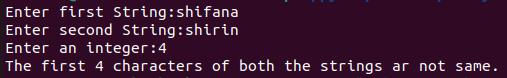
**print(f"The first {n} characters of both the strings are same.")**

**else:**

**print(f"The first {n} characters of both the strings ar not same.")**

**Output:**

****

****

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**PROGRAM NO : 08**

**Aim:**

**Write a program using functions to implement these formulae for permutations and combinations. The Number of permutations of n objects taken r at a time: p(n, r) = n!/(n − r)!. The Number of combinations of n objects taken r at a time is: c(n, r) = n!/(r! ∗ (n − r)!)**

**Source Code:**

**ef factorial(n):**

**if n==0 or n==1:**

**return 1**

**else:**

**return n\*factorial(n-1)**

**def permutation(n,r):**

**return factorial(n)/factorial(n-r)**

**def combination(n,r):**

**return factorial(n)/(factorial(r)\*factorial(n-r))**

**n=int(input("Enter the value of n:"))**

**r=int(input("Enter the value of r:"))**

**print(f"The number of permutaions of {n} objects taken {r} at a time : p({n},{r}) = {n}!/({n}-{r})! = {permutation(n,r)}")**

**print(f"The number of Combinations of n objects taken r at a time : c({n},{r}) = {n}!/({r}! \* ({n}-{r})!) = {combination(n,r)}")**

**Output**