EX-1: STRUCTURED PROGRAMMING

QUESTION 1:

AIM: To print the numbers divisible by 8 and are multiples of 5 between 1000 and2000.

CODE:

```
num=[]
for i in range(1000, 2000):
        if (i%8==0) and (i%5==0):
                num.append(str(i))
print (','.join(num))
```

OUTPUT:

```
1000,1040,1080,1120,1160,1200,1240,1280,1320,1360,1400,1440,1480,1520,1560,
1600,1640,1680,1720,1760,1800,1840,1880,1920,1960
```

RESULT:

Thus, the given program is executed successfully.

QUESTION 2:

AIM: Python program to guess a number between 1 to 9 taking input from the user.

CODE:

```
import random
t_num, g_num = random.randint(1, 10), 0
while t_num != g_num:
g_num = int(input('Guess a number between 1 and 10 until you get it right : '))
print('Well guessed!')
```

OUTPUT:

```
Guess a number between 1 and 10 until you get it right : 5
Well guessed!
```

RESULT:

Thus, the given program is executed successfully.

QUESTION 3:

AIM: To construct the following pattern, using a nested for loop.

CODE:

```
n=5
for i in range(n):
for j in range(i):
print ('* ', end="")
print('')
for i in range(n,0,-1):
for j in range(i):
print('* ', end="")
print('')
```

OUTPUT:

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

RESULT:

Thus, the given program is executed successfully

QUESTION 4:

AIM: To write a program that accepts a word from the user and reverse it.

CODE:

```
w = input("Input a word to reverse: ")
for c in range(len(w) - 1, -1, -1):
```

```
print(w[c], end="")
print("\n")
```

OUTPUT:

```
Input a word to reverse: python
nohtyp
```

RESULT:

Thus, the given program is executed successfully.

QUESTION 5:

AIM: To write a program which takes two digits m (row) and n (column) as input and generates a two-dimensional array.

CODE:

```
r= int(input("Input number of rows: "))
c= int(input("Input number of columns: "))
mult = [[0 for col in range(col_num)] for row in range(row_num)]
for row in range(r):
for col in range(c):
mult[row][col]= row*col
print(mult)
```

OUTPUT:

```
Test Data : Rows = 3, Columns = 4
Expected Result : [[0, 0, 0, 0], [0, 1, 2, 3], [0, 2, 4, 6]]
```

RESULT:

Thus, the given program is executed successfully.

QUESTION 6:

AIM: To write a program that accepts a string and calculates the number of digitsand letters.

CODE:

```
s = input("Input a string")
d=l=0
```

```
for c in s:
if c.isdigit():
d=d+1
elif c.isalpha():
l=l+1
else:
pass
print("Letters", l)
print("Digits", d)
```

OUTPUT:

```
Sample Data: LETTERS 20
Letters 7
Digits 2
```

RESULT:

Thus, the given program is executed successfully.

QUESTION 7:

AIM: To check the validity of password input by users.

CODE:

```
import re
p= input("Input your password")
x = True
while x:
if (len(p)<6 or len(p)>12):
break
elif not re.search("[a-z]",p):
break
elif not re.search("[0-9]",p):
break
elif not re.search("[A-Z]",p):
```

```
break
elif not re.search("[$#@]",p):
break
elif re.search("\s",p):
break
else:
print("Valid Password")
x=False
break
if x:
print("Not a Valid Password")
```

OUTPUT:

```
Input your password srmist@2017
Not a Valid Password
Input your password Srmist@2022
Valid Password
```

RESULT:

Thus, the given program is executed successfully.

QUESTION 8:

AIM: To write a Python program to find numbers between 100 and 400 (bothincluded) where each digit of a number is an even number.

CODE:

```
items = []
for i in range(100, 401):
s = str(i)
if (int(s[0])%2==0) and (int(s[1])%2==0) and (int(s[2])%2==0):
items.append(s)
print( ",".join(items))
```

OUTPUT:

```
200,202,204,206,208,220,222,224,226,228,240,242,244,246,248,260,262,264,266,268,280,282,284,286, 288,400
```

RESULT:

Thus, the given program is executed successfully.

QUESTION 9:

AIM: To convert month name to a number of days.

CODE:

```
print("List of months: January, February, March, April, May, June, July, August,
September, October, November, December")
month_name = input("Input the name of Month: ")
if month_name == "February":
print("No. of days: 28/29 days")
elif month_name in ("April", "June", "September", "November"):
print("No. of days: 30 days")
elif month_name in ("January", "March", "May", "July", "August", "October",
"December"):
print("No. of days: 31 day")
else:
print("Wrong month name")
```

OUTPUT:

```
List of months: January, February, March, April, May, June, July, August,
September, October, November, December
Input the name of Month: May
No. of days: 31 days
```

RESULT:

Thus, the given program is executed successfully.

QUESTION 10:

AIM: To write a Python program to sum of two given integers. However, if the sum isbetween 105 to 200 it will return 200.

CODE:

```
def sum(x, y):
```

```
sum = x + y
if sum in range(105, 200):
return 200
else:
return sum
print(sum(10, 6))
print(sum(10, 2))
print(sum(10, 12))
```

OUTPUT:

```
20
12
22
```

RESULT:

Thus, the given program is executed successfully.

QUESTION 11:

AIM: To construct the given pattern, using a nested loop number.

CODE:

```
for i in range(10):
print(str(i) * i)
```

OUTPUT:

```
999999999
88888888
7777777
666666
55555
4444
333
22
1
```

RESULT:

Thus, the given program is executed successfully.

QUESTION 12:

AIM: To create a histogram from a given list of integers.

CODE:

```
def histogram( items ):
for n in items:
output = ''
times = n
while( times > 0 ):
output += '*'
times = times - 1
print(output)
histogram([2, 3, 6, 5])
```

OUTPUT:

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

RESULT:

Thus, the given program is executed successfully

QUESTION 13:

AIM: To write a Python program that will return true if the two given integer values are equal or their sum or difference is 5.

CODE:

```
def test_number5(x, y):
if x == y or abs(x-y) == 5 or (x+y) == 5:
return True
```

```
else:
return False
print(test_number5(7, 2))
print(test_number5(3, 2))
print(test_number5(2, 2))
print(test_number5(7, 3))
print(test_number5(27, 53))
```

OUTPUT:

```
True
True
True
False
False
```

RESULT:

Thus, the given program is executed successfully.

QUESTION 14:

AIM: To write a Python program to sum of two given integers. However, if the sum isbetween 105 to 200 it will return 200.

CODE:

```
import math
p1 = [4, 0]
p2 = [6, 6]
distance = math.sqrt( ((p1[0]-p2[0])**2)+((p1[1]-p2[1])**2) )
print(distance)
```

RESULT:

Thus, the given program is executed successfully.

QUESTION 15:

AIM: To write a Python program to compute the distance between the points (x1, y1)and (x2, y2).

CODE:

```
def test_distinct(data):
if len(data) == len(set(data)):
return True
else:
return False;
print(test_distinct([1,5,7,9]))
print(test_distinct([2,4,5,5,7,9]))
```

OUTPUT:

```
6.324555320336759
```

RESULT:

Thus, the given program is executed successfully.

QUESTION 17:

AIM: To write a Python program that accept a positive number and subtract from thisnumber the sum of its digits and so on.

CODE:

```
def repeat_times(n):
n_str = str(n)
while (n > 0):
n -= sum([int(i) for i in list(n_str)])
n_str = list(str(n))
return n
print(repeat_times(9))
print(repeat_times(20))
print(repeat_times(110))
print(repeat_times(5674))
```

OUTPUT:

```
0000
```

RESULT:

Thus, the given program is executed successfully.

QUESTION 18:

AIM: To write a Python program to find the digits which are absent in a given mobilenumber.

CODE:

```
def absent_digits(n):
all_nums = set([0,1,2,3,4,5,6,7,8,9])
n = set([int(i) for i in n])
n = n.symmetric_difference(all_nums)
n = sorted(n)
return n
print(absent_digits([9,8,3,2,2,0,9,7,6,3]))
```

OUTPUT:

```
[1, 4, 5]
```

RESULT:

Thus, the given program is executed successfully.

QUESTION 19:

AIM: To write a Python program to reverse the digits of a given number and add it tothe original, If the sum is not a palindrome repeat this procedure

CODE:

```
def rev_number(n):
s = 0
while True:
k = str(n)
if k == k[::-1]:
break
else:
m = int(k[::-1])
n += m
s += 1
return n
print(rev_number(1234))
print(rev_number(1473))
```

OUTPUT:

```
5555
9339
```

RESULT:

Thus, the given program is executed successfully.

QUESTION 20:

AIM: To print the length of the series and the series from the given 3rd term, 3rd lastterm and the sum of a series.

CODE:

```
tn = int(input("Input third term of the series:"))
tltn = int(input("Input 3rd last term:"))
s_sum = int(input("Sum of the series:"))
n = int(2*s_sum/(tn+tltn))
print("Length of the series: ",n)
if n-5==0:
d = (s_sum-3*tn)//6
else:
d = (tltn-tn)/(n-5)
a = tn-2*d
j = 0
print("Series:")
for j in range(n-1):
print(int(a),end=" ")
a+=d
print(int(a),end=" ")
```

OUTPUT:

```
Input third term of the series: 3
Input 3rd last term: 6
Sum of the series: 36
```

```
Length of the series: 8

Series:

1 2 3 4 5 6 7 8
```

RESULT:

Thus, the given program is executed successfully.

EX-2: PROCEDURAL PROGRAMMING

QUESTION 1:

AIM: To print the mirror image of a given string

CODE:

```
str1=input("Enter a string:")
str2=''
ind=-1
for i in str1:
    str2+=str1[ind]
    ind-=1
print(f"The given string = {str1}\nThe Reversed string = {str2}")
```

OUTPUT:

```
Enter a string:python
The given string = python
The Reversed string = nohtyp
```

RESULT:

Thus, the given program is executed successfully.

QUESTION 2:

AIM: Python program to check if two strings are rotationally equal.

CODE:

```
str1=input("Enter   string1:")
str2=input("Enter   string2:")
print("String 1 is:"+str(str1))
print("String 2 is:"+str(str2))
res=False
for i in range(len(str1)):
        if str1[i: ]+str1[ :i]==str2:
            res = True
            break
```

```
if res:
    print("Yes,they are rotationally equal")
else:
    print("No,they are not rotationally equal")
```

OUTPUT:

```
Enter string1:pyi
Enter string2:yip
String 1 is:pyi
String 2 is:yip
Yes,they are rotationally equal
```

RESULT:

Thus, the given program is executed successfully.

QUESTION 3:

AIM: To generate a random binary string of length n.

CODE:

```
import random
n=int(input("Enter length:"))
stri=""
for i in range(n):
    temp = str(random.randint(0,1))
    stri+=temp
print(stri)
```

OUTPUT:

```
Enter length:4
1001
```

RESULT:

Thus, the given program is executed successfully.

QUESTION 4:

AIM: Given a string, remove punctuation and any special characters

CODE:

```
a=input("Enter the string:")
punctuation='''!()-[]{;}:'"\,<>./?@#$%^&*_~'''
for i in a:
    if i in punctuation:
        a=a.replace(i," ")
print(a)
```

OUTPUT:

```
Enter the string:Hi!Hello#You?
Hi Hello You
```

RESULT:

Thus, the given program is executed successfully.

QUESTION 5:

AIM: Write a Python program to compute the element-wise sum of given tuples.

CODE:

```
tuple1=()
list1=[]
n=int(input("Enter the size of tuple1:"))
for i in range(n):
    ele=int(input("Enter the elements for tuple 1:"))
    list1.append(ele)
tuple1=tuple(list1)
tuple2=()
list2=[]
n=int(input("Enter the size of tuple 2:"))
for i in range(n):
    ele=int(input("Enter the elements for tuple 2:"))
    list2.append(ele)
tuple2=tuple(list2)
tuple3=()
list3=[]
n=int(input("Enter the size of tuple 3:"))
for i in range(n):
    ele=int(input("Enter the elements for tuple 3:"))
    list3.append(ele)
tuple3=tuple(list3)
res_tuple=tuple(map(sum,zip(tuple1,tuple2,tuple3)))
print(res_tuple)
```

OUTPUT:

```
Enter the size of tuple1:3
Enter the elements for tuple 1:10
Enter the elements for tuple 1:20
Enter the elements for tuple 1:30
Enter the size of tuple 2:3
Enter the elements for tuple 2:40
Enter the elements for tuple 2:50
Enter the elements for tuple 2:60
Enter the size of tuple 3:3
Enter the elements for tuple 3:70
Enter the elements for tuple 3:80
Enter the elements for tuple 3:90
(120, 150, 180)
```

RESULT:

Thus, the given program is executed successfully.

QUESTION 6:

AIM: Write a Python program to remove an empty tuple(s) from a list of tuples

CODE:

```
List1 = [(), (), ('',), ('c', 'd'), ('m', 'n', 'o')]

List1 = [i for i in List1 if i]

print(List1)
```

OUTPUT:

```
[('',), ('c', 'd'), ('m', 'n', 'o')]
```

RESULT:

Thus, the given program is executed successfully.

QUESTION 7:

AIM: Write a Python program to count the elements in a list until an element is a tuple

CODE:

```
nums=[3,(101,50),(9,),1]

c=0

for i in nums:

    if isinstance(i,tuple):

        c+=1

print(c)
```

OUTPUT:

```
2
```

RESULT:

Thus, the given program is executed successfully.

QUESTION 8:

AIM: Write a Python program to Convert Tuple Matrix to Tuple List

CODE:

```
test=[[(9, 51),(7, 9)],[(11, 1),(22, 19)]]

print("The original list is : " + str(test))
```

```
num = [ele for sub in test for ele in sub]
res = list(zip(*num))
print("The converted tuple list : " + str(res))
```

OUTPUT:

```
The original list is : [[(9, 51), (7, 9)], [(11, 1), (22, 19)]]
The converted tuple list : [(9, 7, 11, 22), (51, 9, 1, 19)]
```

RESULT:

Thus, the given program is executed successfully.

QUESTION 9:

AIM: Write a python program to count unique values in the list

CODE:

```
L = [25,9,0,9,3,0]
print(L)
L1=[]
c=0
for i in L:
    if i not in L1:
        c+=1
        L1.append(i)
print(f"Number of unique items are:{c}",)
```

OUTPUT:

```
[25, 9, 0, 9, 3, 0]
Number of unique items are:4
```

RESULT:

Thus, the given program is executed successfully.

QUESTION 10:

AIM: Python Program to print all Possible Combinations from the three Digits

CODE:

```
def combination(L):
    for i in range(3):
        for j in range(3):
            for k in range(3):
                if (i!=j and j!=k and i!=k):
                    print(L[i], L[j], L[k])
combination([3,5,7])
```

OUTPUT:

```
3 5 7
3 7 5
5 3 7
5 7 3
7 3 5
7 5 3
```

RESULT:

Thus, the given program is executed successfully.

QUESTION 11:

AIM: To write a Python program (using function) to print the even numbers from a given list.

CODE:

```
L=[8,9,7,80,95,2]
def printeven(l):
    for i in l:
        if i%2==0:
            print(i,end=" ")
printeven(L)
```

OUTPUT:

```
8 80 2
```

RESULT:

Thus, the given program is executed successfully.

QUESTION 12:

AIM: Write a Python function (using function) that checks whether a passed string is palindrome or not.

CODE:

```
def palin(s):
    if len(s)<1:
        return True
    else:
        if s[0]==s[-1]:
            return palin(s[1:-1])
        else:
            return False
a=str(input("Enter string:"))
if(palin(a)==True):
    print("String is a palindrome!")
else:
    print("String isn't a palindrome!")
```

OUTPUT:

```
Enter string:ava
String is a palindrome!
```

RESULT:

Thus, the given program is executed successfully.

QUESTION 13:

AIM: To write a Python function (using function) that checks whether a given number is prime or not.

CODE:

```
num=int(input("Enter a number:"))
def isprime(n):
    if n>1:
        for i in range(2,int(n/2)+1):
            if(n%i)==0:
                print(n,"is not a prime number")
                break
        else:
            print(n,"is a prime number")
    else:
        print(n, "is not a prime number")
isprime(num)
```

OUTPUT:

```
Enter a number:7
7 is a prime number
```

RESULT:

Thus, the given program is executed successfully.

EX-3: OBJECT ORIENTED PROGRAMMING

QUESTION 1:

AIM: Write a Python class named SRMIST with six attributes school, dept1, dept2, dept2 and dept3. Add a new attribute specialization and display the entire attribute and their values of the said class. Now remove the dept1 and dept2 attribute and display the entire attribute with values.

CODE:

```
class SRMIST:
    school='SRMIST KTR'
    dept1='CS'
    dept2='CHEM'
    dept3='IOT'
    dept4='ECE'
SRMIST.specialisation='BIG DATA'
var=SRMIST()
print(var.school)
print(var.dept1)
print(var.dept2)
print(var.dept3)
print(var.dept4)
print(var.specialisation)
delattr(SRMIST,'dept1')
delattr(SRMIST,'dept2')
print("After deleting two attributes:")
print(var.school)
print(var.dept3)
print(var.dept4)
print(var.specialisation)
```

OUTPUT:

```
SRMIST KTR
```

```
CS
CHEM
IOT
ECE
BIG DATA
After deleting two attributes:
SRMIST KTR
IOT
ECE
BIG DATA
```

RESULT:

Thus, the given program is executed successfully.

QUESTION 2:

AIM: Write a Python program to crate four empty classes, CTECH, CINTEL, NWC and DSBS. Now create some instances and check whether they are instances of the said classes or not. Also, check whether the said classes are subclasses of the built-in object class or not.

CODE:

```
from ast import Pass

class CTECH:
    pass
class CINTEL:
    pass
class NWC:
    pass
class DSBS:
    pass
o1=CTECH()
```

```
o2=CINTEL()
o3=NWC()
o4=DSBS()
print(“Instance:”)
print(isinstance(o1,CTECH))
print(isinstance(o2,CINTEL))
print(isinstance(o4,NWC))
print(isinstance(o3,NWC))
print(isinstance(o4,DSBS))
print(“Subclass:”)
print(issubclass(CTECH,object))
print(issubclass(CINTEL,object))
print(issubclass(NWC,object))
print(issubclass(DSBS,object))
```

OUTPUT:

```
Instance:
True
True
False
True
True
Subclass:
True
True
True
True
```

RESULT:

Thus, the given program is executed successfully.

QUESTION 3:

AIM: Write a program to print the names of the departments students by creating a Dept class. If no name is passed while creating an object of the Dept class, then the name should be “SCO”, otherwise the name should be equal to the String value passed while creating the object of the Dept class.

CODE:

```
class Dept:
    def _init_(self, *args):
        if len(args) == 1:
            self.dept=args[0]
         elif len(args) == 0:
            self.dept="ECE"
    def deptname(self):
        print(self.dept)
d1=Dept()
d1.deptname()
d2=Dept("EEE")
d2.deptname()
```

OUTPUT:

```
ECE
EEE
```

RESULT:

Thus, the given program is executed successfully.

QUESTION 4:

AIM: Create a class named ‘Rectangle’ with two data members- length and breadth and a function to calculate the area which is ‘length*breadth’. The class has three constructors which are:

1 - having no parameter - values of both length and breadth are assigned

zero.

2 - having two numbers as parameters - the two numbers are assigned as length and breadth respectively.

3 - having one number as parameter - both length and breadth are assigned that number.

Now, create objects of the ‘Rectangle’ class having none, one and two parameters and print their areas.

CODE:

```
class rectangle:
    l=0
    b=0
    def__init__(self,*args):
        if len(args)==2:
            self.l=args[0]
            self.b=args[1]
        elif len(args)==1:
            self.l=args[0]
            self.b=args[0]
        else:
            self.l=0
            self.b=0
    def area(self):
        return self.l*self.b;
r1=rectangle(2,1)
print(r1.area())
r2=rectangle(50)
print(r2.area())
r3=rectangle()
print(r3.area())
```

OUTPUT:

```
2
2500
0
```

RESULT:

Thus, the given program is executed successfully.

QUESTION 5:

AIM: Create a class named 'PrintDT' to print various numbers of different datatypes by creating different functions with the same name 'python_data' having a parameter for each datatype. (example:tuple, list, string)

CODE:

```
class PrintDT:
    def_init_(self,name,age):
        self.name=name
        self.age=age
        print(name,age)
    def python_data(self,hello):
        self.hello=[]
        print(hello)
    def python_data(self,hi):
        self.hi=hi
        print(hi)
    def python_data(self,t):
        self.t=t
        print(t)
    def python_data(self,h):
        self.h=h
        print(h)
bob=PrintDT("Student",18)
```

```
bob.python_data([11,22,33])
bob.python_data("AG")
bob.python_data(("Carrot","Mango", "Orange"))
```

OUTPUT:

```
Student 18
[11, 22, 33]
AG
('Carrot', 'Mango', 'Orange')
```

RESULT:

Thus, the given program is executed successfully.

QUESTION 6:

AIM: A student from SRMIST has his/her money deposited Rs.15000, Rs.30000 and Rs. 40,000 in banks-CUB, HDFC and Indian Bank respectively. We have to print the money deposited by him/her in a particular bank. Create a class named 'Banks_SRMIST' with a function 'getBalance' which returns 0. Make its three subclasses named 'CUB', 'HDFC' and 'Indian_Bank' with a function with the same name 'getBalance' which returns the amount deposited in that particular bank. Call the function 'getBalance' by the object of each of the three banks.

CODE:

```
class Banks_SRMIST:
    def getBalance():
        return 0
class CUB(Banks_SRMIST):
    def getBalance(balance):
         return balance
class HDFC(Banks_SRMIST):
    def getBalance(balance):
        return balance
class Indian_Bank(Banks_SRMIST):
    def getBalance(balance):
        return balance
Banks_SRMIST()
print(CUB.getBalance(1000))
print(HDFC.getBalance(2000))

print(Indian_Bank.getBalance(3000))
```

OUTPUT:

```
1000

2000

3000
```

RESULT:

Thus, the given program is executed successfully.

QUESTION 7:

AIM: Create a Time class and initialize it with hours and minutes.
1. Make a method addTime which should take two time object and add them. E.g.- (2 hour and 50 min)+(1 hr and 20 min) is (4 hr and 10 min)
2. Make a method displayTime which should print the time.
3. Make a method DisplayMinute which should display the total minutes in the Time. E.g.- (1 hr 2 min) should display 62 minute.

CODE:

```
class Time():

    def_init_(self, hours, mins):

        self.hours = hours

        self.mins = mins

    def addTime(t1, t2):

        t3 = Time(0,0)

        if t1.mins+t2.mins > 60:

            t3.hours = (t1.mins+t2.mins)//60

        t3.hours = t3.hours+t1.hours+t2.hours

        t3.mins = (t1.mins + t2.mins) % 60

        return t3

    def displayTime(self):

        print("Time is",self.hours,"hours and",self.mins,"minutes.")

    def displayMinute(self):

        print ((self.hours*60)+self.mins)
```

```
a = Time(1,10)
b = Time(2,20)
c = Time.addTime(a,b)
c.displayTime()
c.displayMinute()
```

OUTPUT:

```
Time is 3 hours and 30 minutes.
210
```

RESULT:

Thus, the given program is executed successfully.

QUESTION 8:

AIM: Write a program to print the area and perimeter of a triangle having sides of 3, 4 and 5 units by creating a class named 'Triangle' with a function to print the area and perimeter.

CODE:

```
class Triangle:
    def findPerimeter(self,s1,s2,s3):
        return (s1+s2+s3)
    def findArea(self,s1,s2,s3):
        s=(s1+s2+s3)/2
        return ((s*(s-s1)*(s-s2)*(s-s3))**0.5)
s1=float(input("Enter first side of the triangle:"))
s2=float(input("Enter second side of the triangle:"))
s3=float(input("Enter third side of the triangle:"))
u=Triangle()
print("Perimeter of the triangle:{0:.2f}".format(u.findPerimeter(s1,s2,s3)))
print("Area of the triangle is:{0:.2f}".format(u.findArea(s1,s2,s3)))
```

OUTPUT:

```
Enter first side of the triangle:4
Enter second side of the triangle:8
Enter third side of the triangle:10
```

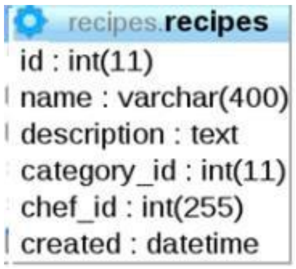
```
Perimeter of the triangle:22.00

Area of the triangle is:15.20
```

RESULT: Thus, the given program is executed successfully

EX-4: DECLARATIVE PROGRAMMING

QUESTION 1:

AIM: Create the below table and execute the insert, update and the below select statements.



recipes.recipes

id : int(11)
name : varchar(400)
description : text
category_id : int(11)
chef_id : int(255)
created : datetime

i) Write a query to display the total number of recipes available with the description “Chinese”

ii) Write a query to display the id, name of the recipes with chef_id 'BL000002'.

iii) Write a query to display the description of the recipes whose name begins with 'P'.

CODE:

```
import sqlite3

db=sqlite3.connect('recipes.db')

curs=db.cursor()

curs.execute('CREATE TABLE recipes(id int,name varchar(400),description text,category_id int,chef_id int,created datetime)')

print("Table is ready")

curs.execute('''INSERT INTO recipes (id,name,description,category_id,chef_id,created)

            VALUES(1,'Kiami','Chinese',11,'BL000002','2022-4-4'),

              (2,'Caio','Korean',12,'BL000001','2022-4-4'),

              (3,'Chingua','Japanese',13,'BL000003','2022-4-4'),

              (4,'Puniket','Indian',14,'BL000004','2022-4-4'),

              (5,'Shin','Chinese',15,'BL000002','2022-4-4')''')

curs.execute("SELECT count(id) FROM recipes where description='Chinese'")

res=curs.fetchall()

for i in res:
```

```
print(i)
```

```
curs.execute("SELECT id,name FROM recipes where chef_id='BL000002'")
res=curs.fetchall()
for i in res:
    print(i)
curs.execute("SELECT * FROM recipes where name like 'P%'")
res=curs.fetchall()
for i in res:
   print(i)
 db.close()
```

OUTPUT:

```
(2,)
(1, 'Kiami')
(5, 'Shin')
(4, 'Puniket', 'Indian', 14, 'BL000004', '2022-4-4')
```

RESULT:

Thus, the given program is executed successfully.

QUESTION 2:

AIM: Create a table movie of the below structure and assume data types.Movie_ID, Movie_Name, Genre, Language, Rating ,Do the following queries

a. Update the movies rating by 10% and display it

b. Delete the movies with movie_id 102

c. Select movies whose rating is more than 3

CODE:

```
import sqlite3
db=sqlite3.connect('mydatabase.db')
curs=db.cursor()
curs.execute("""CREATE TABLE IF NOT EXISTS Movie (
            Movie_ID VARCHAR(255),
            Movie_Name VARCHAR(255),
        Genre VARCHAR(255),
        Language VARCHAR(255),
        Rating FLOAT);""")
curs.execute("""INSERT INTO Movie VALUES ("101","No time to die","Mystery","English","8");""")
curs.execute("""INSERT INTO Movie VALUES ("102","POTC","Adventure","English","9");""")
curs.execute("""INSERT INTO Movie VALUES ("103","Jurassic Park","Thriller","English","9.2");""")
curs.execute("""INSERT INTO Movie VALUES ("107","Mission Impossible","Thriller","English","7.5");""")
curs.execute("""INSERT INTO Movie VALUES ("002","The Flopped","drama","English","2.5");""")
print("Original data")
curs.execute("SELECT * FROM Movie")
res=curs.fetchall()
for i in res:
    print(i)
print("")
curs.execute("""UPDATE Movie SET Rating = ((Rating*110)/100);""")
print("Printing data after updating rating to 10%")
curs.execute("SELECT * FROM Movie")
res=curs.fetchall()
for i in res:
    print(i)
print("")
curs.execute("""DELETE from Movie where Movie_ID = 102 """)
print("Displaying data after deleting 102 Movie_ID")
curs.execute("SELECT * FROM Movie")
res=curs.fetchall()
for i in res:
    print(i)
print("")
print("Displaying data having rating value greater than 3")
curs.execute("SELECT * FROM Movie WHERE Rating>3")
```

```
res=curs.fetchall()
for i in res:
    print(i)
print("")
db.commit()
db.close()
```

OUTPUT:

```
Original data
('101', 'No time to die', 'Mystery', 'English', 8.0)
('102', 'POTC', 'Adventure', 'English', 9.0)
('103', 'Jurassic Park', 'Thriller', 'English', 9.2)
('107', 'Mission Impossible', 'Thriller', 'English', 8.5)
('108', 'The Flopped', 'drama', 'English', 2.5)

Printing data after updating rating to 10%
('101', 'No time to die', 'Mystery', 'English', 8.8)
('102', 'POTC', 'Adventure', 'English', 9.9)
('103', 'Jurassic Park', 'Thriller', 'English', 10.12)
('107', 'Mission Impossible', 'Thriller', 'English', 9.35)
('108', 'The Flopped', 'drama', 'English', 2.75)

Displaying data after deleting 102 Movie_ID
('101', 'No time to die', 'Mystery', 'English', 8.8)
('103', 'Jurassic Park', 'Thriller', 'English', 10.12)
('107', 'Mission Impossible', 'Thriller', 'English', 9.35)
('108', 'The Flopped', 'drama', 'English', 2.75)

Displaying data having rating value greater than 3
('101', 'No time to die', 'Mystery', 'English', 8.8)
```

```
('103', 'Jurassic Park', 'Thriller', 'English', 10.12)
('107', 'Mission Impossible', 'Thriller', 'English', 9.35)
```

RESULT:

Thus, the given program is executed successfully.

QUESTION 3:

AIM: Create a course database with the following fields Product(ID, Prod_name,

Supplier_id,Unit_price,Package,OrderID),OrderItem(ID,Order_id,Product_id,Unit_price, Quantity) using Foreign key

d. Display the total quantity of every product in the stock

e. Sort the Unit_price based on the supplier_id

f. Display the Product_name along with order_id and supplier_id

CODE:

```
import sqlite3
db=sqlite3.connect("mydatabase1.db")
curs=db.cursor()
db.execute("PRAGMA foreign_keys = ON")
curs.execute("""CREATE TABLE IF NOT EXISTS Product(
        ID VARCHAR(255) PRIMARY KEY,
        Prod_name  VARCHAR(255),
        Supplier_id VARCHAR(255),
        Unit_price INT,
        Package INT,
        Order_ID VARCHAR(255)
        );""")
curs.execute("""INSERT OR IGNORE INTO Product VALUES('id1','pid1','1','500','100','Oid1');""")
curs.execute("""INSERT OR IGNORE INTO Product VALUES('id2','pid2','2','1000','200','Oid2');""")
curs.execute("""INSERT OR IGNORE INTO Product VALUES('id3','pid3','3','1500','300','Oid3');""")
curs.execute("""INSERT OR IGNORE INTO Product VALUES('id4','pid4','4','2000','400','Oid4');""")
curs.execute("""CREATE TABLE IF NOT EXISTS OrderItem(
        ID VARCHAR(255),
        Order_id VARCHAR(255),
        Product_ID VARCHAR(255),
        Unit_price INT ,
        Quantity INT,
        FOREIGN KEY (ID) REFERENCES Product(ID)
        );""")
curs.execute("""INSERT OR IGNORE INTO OrderItem VALUES('id1','oid1','pid1','500','10');""")
curs.execute("""INSERT OR IGNORE INTO OrderItem VALUES('id2','oid2','pid2','1000','20');""")
curs.execute("""INSERT OR IGNORE INTO OrderItem VALUES('id3','oid3','pid3','1500','30');""")
curs.execute("""INSERT OR IGNORE INTO OrderItem VALUES('id4','oid4','pid4','2000','40');""")
print("Displaying total quantity of each product")
res=curs.execute("SELECT Quantity FROM OrderItem where ID = 'id1'")
for i in res:
  print(i)
res=curs.execute("SELECT Quantity FROM OrderItem where ID = 'id2'")
for i in res:
  print(i)
res=curs.execute("SELECT Quantity FROM OrderItem where ID = 'id3'")
for i in res:
  print(i)
res=curs.execute("SELECT Quantity FROM OrderItem where ID = 'id4'")
for i in res:
  print(i)
print("")
print("Displaying unit price based on supply id")
res=curs.execute("SELECT Unit_price FROM Product")
for i in res:
  print(i)
print("")
print("Displaying Product name along with order id and supplier id")
```

```
res=curs.execute("SELECT Prod_name ,Order_ID ,Supplier_id FROM Product")
for i in res:
    print(i)
db.close()
```

OUTPUT:

```
Displaying total quantity of each product
(10,)
(20,)
(30,)
(40,)

Displaying unit price based on supply id
(500,)
(1000,)
(1500,)
(2000,)

Displaying Product name along with order id and supplier id
('pid1', 'Oid1', '1')
('pid2', 'Oid2', '2')
('pid3', 'Oid3', '3')
('pid4', 'Oid4', '4')
```

RESULT:

Thus, the given program is executed successfully.

QUESTION 4:

AIM: Write a SQL lite3 statement to create a table named as job including columns job_id,job_title,Min-salary,Max_salary.job_id column does not contain any duplicate value at the time of insertion.

CODE:

```
import sqlite3
db=sqlite3.connect('mydatabase2.db')
```

```
curs=db.cursor()

curs.execute("""CREATE TABLE IF NOT EXISTS Job(

            job_id VARCHAR(255),

            job_title VARCHAR(255),

            Min_salary INT,

            Max_salary INT);""")

curs.execute("""INSERT INTO Job VALUES ("fsd01","Full Stack Developer","100000","1000000");""" )

curs.execute("""INSERT INTO Job VALUES ("gd02","Game Developer","30000","300000");""")

curs.execute("""INSERT INTO Job VALUES ("an03","Animation Developer","50000","500000");""" )

curs.execute("SELECT * FROM Job")

res=curs.fetchall()

for i in res:

    print(i)

print("")
```

OUTPUT:

```
('fsd01', 'Full Stack Developer', 100000, 1000000)

('gd02', 'Game Developer', 30000, 300000)

('an03', 'Animation Developer', 50000, 500000)
```

RESULT:

Thus, the given program is executed successfully.

QUESTION 5:

AIM: Write a SQL lite3 statement to create a table names as job_history including columns employee_id, start_date, end_date, job_id and department_id and make sure that, the employee_id column does not contain any duplicate value at the time of insertion and the foreign key column job_id contain only those values which are exists in the jobs table.

CODE:

```
import sqlite3

db=sqlite3.connect("mydatabase3.db")

curs=db.cursor()

db.execute("PRAGMA foreign_keys = ON")

curs.execute("""CREATE TABLE IF NOT EXISTS jobs(
            Job_id VARCHAR(255) PRIMARY KEY
            );""")
curs.execute("""INSERT OR IGNORE INTO jobs VALUES ('j1');""")
curs.execute("""INSERT OR IGNORE INTO jobs VALUES('j2');""")
curs.execute("""INSERT OR IGNORE INTO jobs VALUES('j3');""")
curs.execute("""CREATE TABLE IF NOT EXISTS job_history(
            employeeid INTEGER NOT NULL,
            startdate DATE NOT NULL,
            enddate DATE NOT NULL,
            job_id VARCHAR(255) NOT NULL,
            departmentid INTEGER NOT NULL,
            FOREIGN KEY (job_id) REFERENCES jobs(Job_id)
            );""")
curs.execute("""INSERT OR IGNORE INTO job_history VALUES('1','2000-02-20','2010-02-20','j1','01');""")
curs.execute("""INSERT OR IGNORE INTO job_history VALUES('2','2005-10-09','2016-05-10','j2','02');""")
curs.execute("""INSERT INTO job_history VALUES('5','1990-05-11','1995-06-19','j1','10');""")
curs.execute("""INSERT OR IGNORE INTO job_history VALUES('23','1998-02-15','2011-10-12','j3','05');""")
curs.execute("SELECT * FROM job_history where job_id='j1'")
res=curs.fetchall()
for i in res:
   print(i)
 db.close()
```

OUTPUT:

```
(1, '2000-02-20', '2010-02-20', 'j1', 1)
(5, '1990-05-11', '1995-06-19', 'j1', 10)
```

RESULT:

Thus, the given program is executed successfully.

EX-5: EVENT DRIVEN PROGRAMMING

QUESTION 1:

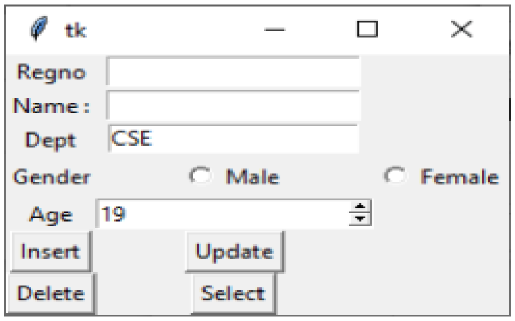
AIM: To create the given design.

CODE:

```
from tkinter import *
root=Tk()

root.geometry=('500x500')
root.title('Student Registration')
label_0=Label(root, text='Regno')
label_0.grid(row=1,column=1)
entry_1=Entry(root)
entry_1.grid(row=1,column=2)

label_1=Label(root, text='Name:')
label_1.grid(row=2,column=1)
entry_2=Entry(root)
entry_2.grid(row=2,column=2)

label_2=Label(root, text='Dept')
label_2.grid(row=3,column=1)
entry_3=Entry(root)
entry_3.grid(row=3,column=2)

var=IntVar()
label_3=Label(root, text='Gender')
label_3.grid(row=4,column=1)
rb1=Radiobutton(root,text='Male',variable=var,value=1)
rb1.grid(row=4,column=2)
rb2=Radiobutton(root,text='Female',variable=var,value=2)
```

```
rb2.grid(row=4,column=3)

label_4=Label(root, text='Age')
label_4.grid(row=5,column=1)
entry_5=Spinbox(root, from_=1, to=80)
entry_5.grid(row=5,column=2)

b1=Button(root, text='Insert')
b1.grid(row=6,column=1)

b1=Button(root, text='Update')
b1.grid(row=6,column=2)

b1=Button(root, text='Delete')
b1.grid(row=7,column=1)

b1=Button(root, text='Select')
b1.grid(row=7,column=2)

root.mainloop()
```

OUTPUT:



RESULT:

Thus, the given program is executed successfully.

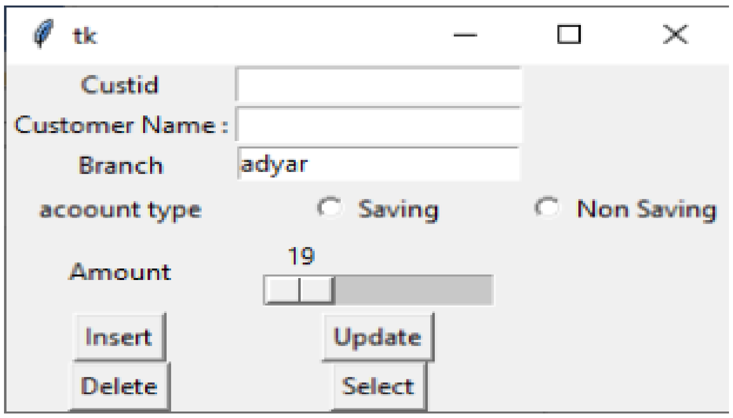
QUESTION 2:

AIM: To create the given design.

CODE:

```
from tkinter import *
root=Tk()

root.geometry=('500x500')
root.title('Banking')
label_0=Label(root, text='Custid')
label_0.grid(row=1,column=1)
entry_1=Entry(root)
entry_1.grid(row=1,column=2)

label_1=Label(root, text='Customer Name:')
label_1.grid(row=2,column=1)
entry_2=Entry(root)
entry_2.grid(row=2,column=2)

label_2=Label(root, text='Branch')
label_2.grid(row=3,column=1)
entry_3=Entry(root)
entry_3.grid(row=3,column=2)

var=IntVar()
label_3=Label(root, text='Account Type')
label_3.grid(row=4,column=1)
rb1=Radiobutton(root,text='Saving',variable=var,value=1)
rb1.grid(row=4,column=2)
rb2=Radiobutton(root,text='Non Saving',variable=var,value=2)
rb2.grid(row=4,column=3)
```

```
label_4=Label(root, text='Amount')

label_4.grid(row=5,column=1)

entry_5=Scale(root, variable=var,orient=HORIZONTAL)

entry_5.grid(row=5,column=2)


b1=Button(root, text='Insert')

b1.grid(row=6,column=1)


b1=Button(root, text='Update')

b1.grid(row=6,column=2)


b1=Button(root, text='Delete')

b1.grid(row=7,column=1)


b1=Button(root, text='Select')

b1.grid(row=7,column=2)


root.mainloop()
```
OUTPUT:



RESULT:

Thus, the given program is executed successfully.

QUESTION 3:

AIM: To create the given design.

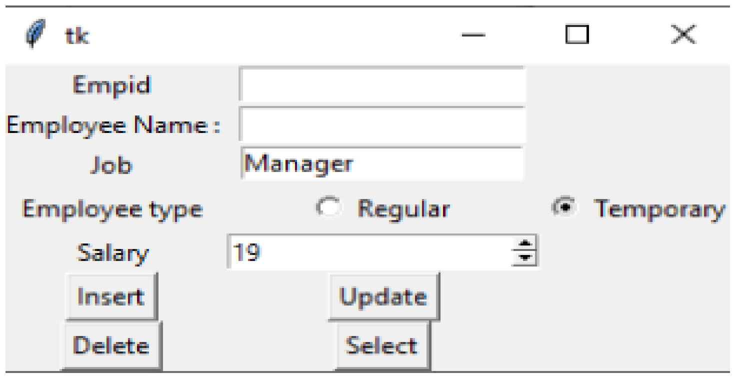
CODE:

```
from tkinter import *

root=Tk()

root.geometry=('500x500')

root.title('Employee')

label_0=Label(root, text='Empid')

label_0.grid(row=1,column=1)

entry_1=Entry(root)

entry_1.grid(row=1,column=2)



label_1=Label(root, text='Employee Name:')

label_1.grid(row=2,column=1)

entry_2=Entry(root)

entry_2.grid(row=2,column=2)



label_2=Label(root, text='Job')

label_2.grid(row=3,column=1)

entry_3=Entry(root)

entry_3.grid(row=3,column=2)
```

```
var=IntVar()

label_3=Label(root, text='Employee type')

label_3.grid(row=4,column=1)

rb1=Radiobutton(root,text='Regular',variable=var,value=1)

rb1.grid(row=4,column=2)

rb2=Radiobutton(root,text='Temporary',variable=var,value=2)

rb2.grid(row=4,column=3)


label_4=Label(root, text='Salary')

label_4.grid(row=5,column=1)

entry_5=Spinbox(root, from_=1, to=10000)

entry_5.grid(row=5,column=2)


b1=Button(root, text='Insert')

b1.grid(row=6,column=1)


b1=Button(root, text='Update')

b1.grid(row=6,column=2)


b1=Button(root, text='Delete')

b1.grid(row=7,column=1)
```

```
b1=Button(root, text='Select')

b1.grid(row=7,column=2)



root.mainloop()
```

OUTPUT:



tk

Empid
Employee Name :
Job Manager
Employee type ( ) Regular (●) Temporary
Salary 19

\

RESULT:

Thus, the given program is executed successfully.

QUESTION 4:

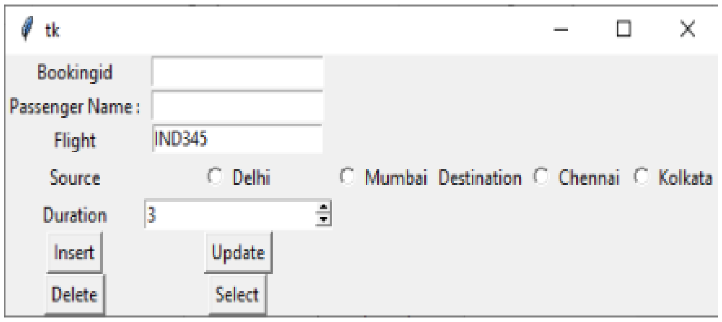
AIM: To create the given design.

CODE:

```
from tkinter import *
root=Tk()

root.geometry=('500x500')
root.title('Flyer')
label_0=Label(root, text='Bookingid')
label_0.grid(row=1,column=1)
entry_1=Entry(root)
entry_1.grid(row=1,column=2)

label_1=Label(root, text='Passenger Name:')
label_1.grid(row=2,column=1)
entry_2=Entry(root)
entry_2.grid(row=2,column=2)

label_2=Label(root, text='Flight')
label_2.grid(row=3,column=1)
entry_3=Entry(root)
entry_3.grid(row=3,column=2)

var=IntVar()
label_3=Label(root, text='Source')
label_3.grid(row=4,column=1)
rb1=Radiobutton(root,text='Delhi',variable=var,value=1)
rb1.grid(row=4,column=2)
rb2=Radiobutton(root,text='Mumbai',variable=var,value=2)
rb2.grid(row=4,column=3)
```

```
rb1=Radiobutton(root,text='Chennai',variable=var,value=1)
rb1.grid(row=4,column=4)
rb2=Radiobutton(root,text='Kolkata',variable=var,value=2)
rb2.grid(row=4,column=5)

label_4=Label(root, text='Duration')
label_4.grid(row=5,column=1)
entry_5=Spinbox(root, from_=1, to=80)
entry_5.grid(row=5,column=2)

b1=Button(root, text='Insert')
b1.grid(row=6,column=1)

b1=Button(root, text='Update')
b1.grid(row=6,column=2)

b1=Button(root, text='Delete')
b1.grid(row=7,column=1)

b1=Button(root, text='Select')
b1.grid(row=7,column=2)

root.mainloop()
```

OUTPUT:



RESULT:

Thus, the given program is executed successfully.

QUESTION 5:

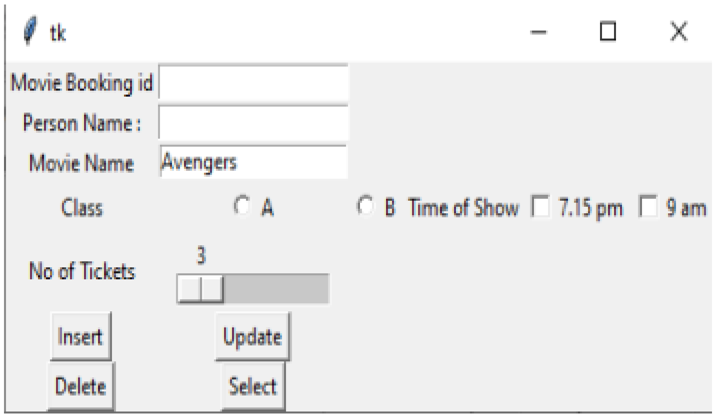
AIM: To create the given design.

Code:

```
from tkinter import *

root=Tk()

root.geometry=('500x500')

root.title('Movie Booking Ticket')

label_0=Label(root, text='Movie Booking id')

label_0.grid(row=1,column=1)

entry_1=Entry(root)

entry_1.grid(row=1,column=2)

label_1=Label(root, text='Person Name:')

label_1.grid(row=2,column=1)

entry_2=Entry(root)

entry_2.grid(row=2,column=2)

label_2=Label(root, text='Movie Name')

label_2.grid(row=3,column=1)

entry_3=Entry(root)

entry_3.grid(row=3,column=2)

var=IntVar()

CheckVar1 = IntVar()

CheckVar2 = IntVar()

label_3=Label(root, text='Class')

label_3.grid(row=4,column=1)

rb1=Radiobutton(root,text='A',variable=var,value=1)
```

```
rb1.grid(row=4,column=2)
rb2=Radiobutton(root,text='B',variable=var,value=2)
rb2.grid(row=4,column=3)
label_3_1=Label(root, text='Time of Show')
label_3_1.grid(row=4,column=4)
rb1=Checkbutton(root,text='7.15 pm',variable=CheckVar1, onvalue=1, offvalue=0)
rb1.grid(row=4,column=5)
rb2=Checkbutton(root,text='9 am',variable=CheckVar2, onvalue=1, offvalue=0)
rb2.grid(row=4,column=6)
label_4=Label(root, text='Duration')
label_4.grid(row=5,column=1)
entry_5=Scale(root, from_=3, to=100, orient='horizontal')
entry_5.grid(row=5,column=2)
b1=Button(root, text='Insert')
b1.grid(row=6,column=1)
b1=Button(root, text='Update')
b1.grid(row=6,column=2)
b1=Button(root, text='Delete')
b1.grid(row=7,column=1)
b1=Button(root, text='Select')
b1.grid(row=7,column=2)
root.mainloop()
```

Output:



RESULT:

Thus, the given program is executed successfully.

QUESTION 6:

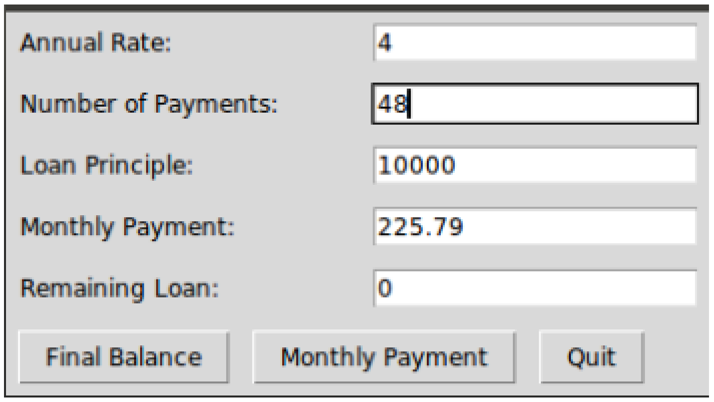
AIM: To create the given design.

CODE:

```
from tkinter import *
root=Tk()

root.geometry=('500x500')
root.title('Customer Loan Details')
label_0=Label(root, text='Annual Rate:')
label_0.grid(row=1,column=1)
entry_1=Entry(root)
entry_1.grid(row=1,column=2)

label_1=Label(root, text='Number of Payments:')
label_1.grid(row=2,column=1)
entry_2=Entry(root)
entry_2.grid(row=2,column=2)

label_2=Label(root, text='Loan Principle:')
label_2.grid(row=3,column=1)
entry_3=Entry(root)
entry_3.grid(row=3,column=2)

label_2=Label(root, text='Monthly Payment:')
label_2.grid(row=4,column=1)
entry_3=Entry(root)
```

```
entry_3.grid(row=4,column=2)
label_2=Label(root, text='Remaining Loan:')
label_2.grid(row=5,column=1)
entry_3=Entry(root)
entry_3.grid(row=5,column=2)
b1=Button(root, text='Final Balance')
b1.grid(row=6,column=1)
b1=Button(root, text='Monthly Payment')
b1.grid(row=6,column=2)
b1=Button(root, text='Quit')
b1.grid(row=6,column=3)
root.mainloop()
```

OUTPUT:



Annual Rate: 4
Number of Payments: 48
Loan Principle: 10000
Monthly Payment: 225.79
Remaining Loan: 0

RESULT:

Thus, the given program is executed successfully.

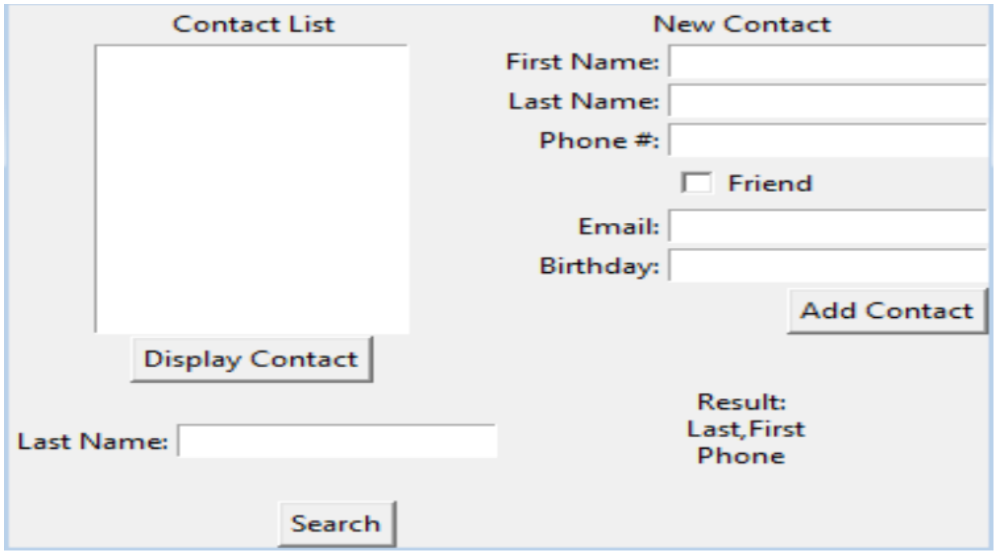
QUESTION 7:

AIM: To create the given design.

CODE:

```
from tkinter import *
root = Tk()
Lname = StringVar()
lname, fname = StringVar(), StringVar()
email = StringVar()
bday = StringVar()
Label(root, text="Contact List", width=25, justify=CENTER).grid(row=0,
                                                    column=0)
contactList = Listbox(root, width=25, height=8).grid(row=1, column=0,
                                          rowspan=4)
Button(root, text="Display Contact", width=20, height=1,
    justify=CENTER).grid(row=5, column=0, pady=8)
name = Label(root, text="Last Name ", width=15, justify=LEFT).grid(row=6,
                                                    column=0)
Entry(root, textvariable=Lname, width=25).grid(row=6, column=1)
 search = Button(root, text="Search", width=15, justify=CENTER).grid(row=7,column=1, pady=15)
Label(root, text="New Contact : ", width=25, justify=CENTER).grid(row=0,
                                                    column=2, padx=8)
fn = Label(root, text="First Name : ", width=15, justify=LEFT,
      anchor="e").grid(row=1, column=2, padx=8)
Entry(root, textvariable=fname, width=25).grid(row=1, column=3)
ln = Label(root, text="Last Name : ", width=15, justify=LEFT,
          anchor="e").grid(row=2, column=2, padx=8)
Entry(root, textvariable=lname, width=25).grid(row=2, column=3)
phn = Label(root, text="Phone # : ", width=15, justify=LEFT,
           anchor="e").grid(row=3, column=2, padx=8)
Entry(root, textvariable=phn, width=25).grid(row=3, column=3)
emailEl = Label(root, text="Email : ", width=15, justify=LEFT,
               anchor="e").grid(row=4, column=2, padx=8)
Entry(root, textvariable=email, width=25).grid(row=4, column=3)
bdayEl = Label(root, text="Birthday : ", width=15, justify=LEFT,
              anchor="e").grid(row=5, column=2, padx=8)
Entry(root, textvariable=bday, width=25).grid(row=5, column=3)
Button(root, text="Add Contact", width=15, justify=RIGHT).grid(row=6,
                                                 column=3, sticky="e")
Label(root, text="Result:\nLast, First\nPhone").grid(row=8, column=3,
                                          sticky="sw")
root.mainloop()
```

OUTPUT:



RESULT:

Thus, the given program is executed successfully.

QUESTION 8:

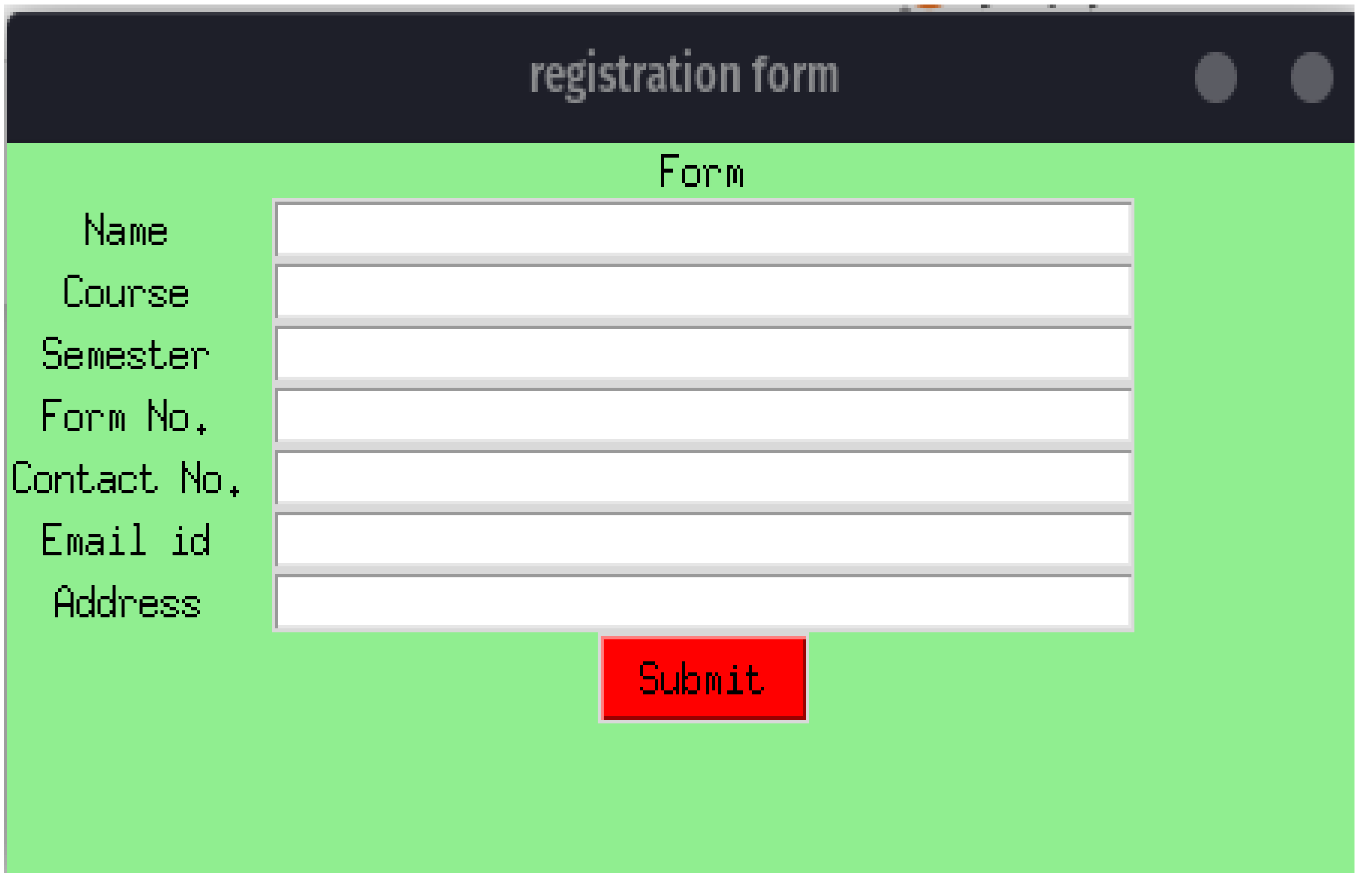
AIM: To create the given design.

CODE:

```
 import tkinter as tk
from tkinter import messagebox

root=tk.Tk()
root.geometry("450x200")
root.configure(background="lightgreen")
root.title('registration form')
a=tk.Label(root, text='Form',background="lightgreen").grid(row=0,column=1)
l1 = tk.Label(root,text = "Name ", background="lightgreen").grid(row = 1, column = 0)
t=tk.Entry(root,width=40).grid(row=1,column=1)
l2=tk.Label(root,text="Course ", background="lightgreen").grid(row = 2, column = 0)
t=tk.Entry(root,width=40).grid(row=2,column=1)
l3=tk.Label(root,text="Semester ", background="lightgreen").grid(row = 3, column = 0)
t=tk.Entry(root, width=40).grid(row=3,column=1)
l4 = tk.Label(root,text ="Form No. ",background="lightgreen").grid(row = 4, column = 0)
t=tk.Entry(root, width=40).grid(row=4,column=1)
l5=tk.Label(root,text="Contact No. ",background="lightgreen").grid(row = 5, column = 0)
t=tk.Entry(root, width=40).grid(row=5,column=1)
l6=tk.Label(root,text="Email id ",background="lightgreen").grid(row = 6, column = 0)
t=tk.Entry(root,width=40).grid(row=6,column=1)
l6=tk.Label(root,text="Address ",background="lightgreen").grid(row = 7, column = 0)
t=tk.Entry(root,width=40).grid(row=7,column=1)
button=tk.Button(root, text='Submit', width=6, background="red").grid(row=8,column=1)
```

```
root.mainloop()
```

OUTPUT:


registration form

Form

Name

Course

Semester

Form No.

Contact No.

Email id

Address

Submit

RESULT:

Thus, the given program is executed successfully.

QUESTION 9:

AIM: To create the given design.

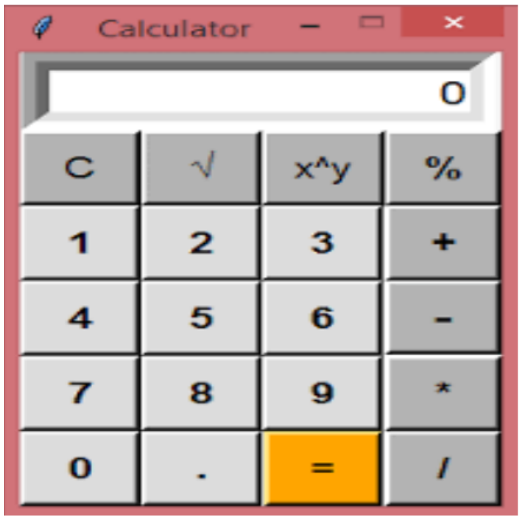
CODE:

```
from tkinter import *
expression = ""
def press(num):
    global expression
    expression = expression + str(num)
    equation.set(expression)

def equalpress():
    try:
        global expression
        total = str(eval(expression))
        equation.set(total)
        expression = ""
    except:
        equation.set(" error ")
        expression = ""

def clear():
    global expression
    expression = ""
    equation.set("0")

if __name__ == "__main__":
```

```
gui = Tk()
gui.title("Calculator")
gui.geometry("270x150")
equation = StringVar()
expression_field = Entry(gui, textvariable=equation)
expression_field.grid(columnspan=4, ipadx=70)
clrscr = Button(gui, text=' C ', fg='black', bg='grey',
            command=clear, height=1, width=7)
clrscr.grid(row=2, column=0)

squareRoot = Button(gui, text=' √ ', fg='black', bg='grey',
            command=lambda: press('√'), height=1, width=7)
squareRoot.grid(row=2, column=1)

exponent = Button(gui, text=' x^y ', fg='black', bg='grey',
            command=lambda: press('^'), height=1, width=7)
exponent.grid(row=2, column=2)

percent = Button(gui, text=' % ', fg='black', bg='grey',
            command=lambda: press('%'), height=1, width=7)
percent.grid(row=2, column=3)

num1 = Button(gui, text=' 1 ', fg='black',
            command=lambda: press(1), height=1, width=7)
num1.grid(row=3, column=0)

num2 = Button(gui, text=' 2 ', fg='black',
            command=lambda: press(2), height=1, width=7)
num2.grid(row=3, column=1)

num3 = Button(gui, text=' 3 ', fg='black',
                command=lambda: press(3), height=1, width=7)
num3.grid(row=3, column=2)

plus = Button(gui, text=' + ', fg='black', bg='grey',
                command=lambda: press('+'), height=1, width=7)
plus.grid(row=3, column=3)

num4 = Button(gui, text=' 4 ', fg='black',
                command=lambda: press(4), height=1, width=7)
num4.grid(row=4, column=0)

num5 = Button(gui, text=' 5 ', fg='black',
                command=lambda: press(5), height=1, width=7)
num5.grid(row=4, column=1)

num6 = Button(gui, text=' 6 ', fg='black',
            command=lambda: press(6), height=1, width=7)
num6.grid(row=4, column=2)

minus = Button(gui, text=' - ', fg='black', bg='grey',
            command=lambda: press("-"), height=1, width=7)
minus.grid(row=4, column=3)

num7 = Button(gui, text=' 7 ', fg='black',
                command=lambda: press(7), height=1, width=7)
num7.grid(row=5, column=0)

num8 = Button(gui, text=' 8 ', fg='black',
                command=lambda: press(8), height=1, width=7)
num8.grid(row=5, column=1)
```

```
    num9 = Button(gui, text=' 9 ', fg='black',
                 command=lambda: press(9), height=1, width=7)
    num9.grid(row=5, column=2)

    multiply = Button(gui, text=' * ', fg='black', bg='grey',
                 command=lambda: press('*'), height=1, width=7)
    multiply.grid(row=5, column=3)

    num0= Button(gui, text=' 0 ', fg='black',
                   command=lambda: press(0), height=1, width=7)
    num0.grid(row=6, column=0)

    decimal= Button(gui, text=' . ', fg='black',
                   command=lambda: press('.'), height=1, width=7)
    decimal.grid(row=6, column=1)

    equals= Button(gui, text=' = ', fg='black', bg='orange',
                   command=equalpress, height=1, width=7)
    equals.grid(row=6, column=2)

    divide= Button(gui, text=' / ', fg='black', bg='grey',
                   command=lambda: press('/'), height=1, width=7)
    divide.grid(row=6, column=3)
    gui.mainloop()
```

OUTPUT:



RESULT: Thus, the given program is executed successfully.

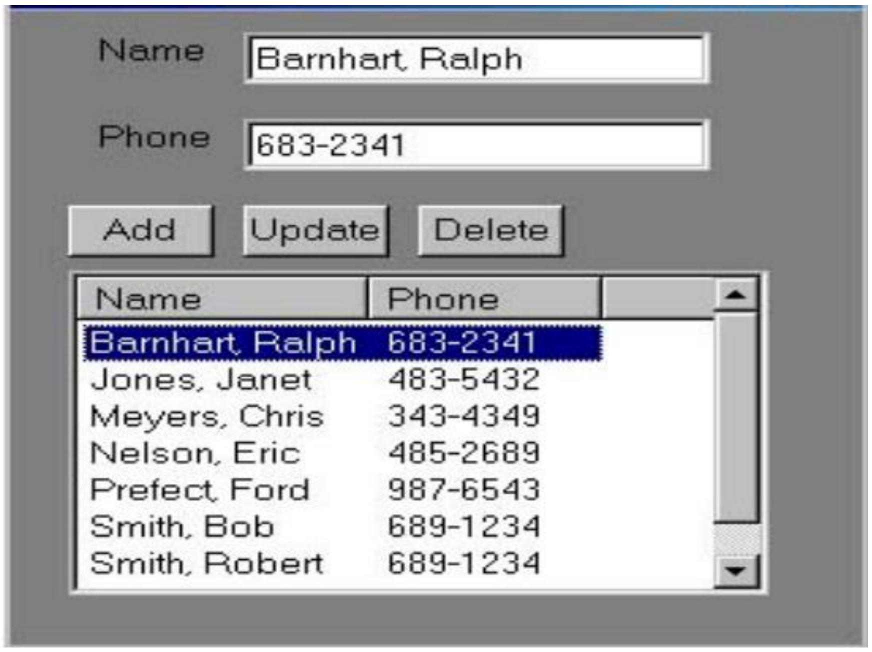
QUESTION 10:

AIM: To create the given design.

CODE:

```
from tkinter import *

from tkinter import ttk

root = Tk()

root.title('Phone List')

root.geometry("450x300")

l1=Label(root,text = "Name: ").place(x=50,y=10)

t=Entry(root,width=35).place(x=130,y=10)

l2=Label(root,text="Phone ").place(x=50,y=40)

t=Entry(root,width=35).place(x=130,y=40)

button=Button(root, text='Add', width=8).place(x=40,y=70)

button=Button(root, text='Update', width=10).place(x=130,y=70)

button=Button(root, text='Delete', width=10).place(x=230,y=70)

style = ttk.Style()

style.theme_use("default")

frame = Frame(root,width=10,height=100)

frame.pack()
frame.place(x=20,y=110)
tv = ttk.Treeview(frame, columns=(1, 2,3), show='headings', height=7)
tv.pack(side=LEFT)
tv.heading(1, text="Name",anchor="w")
tv.heading(2, text="Phone",anchor="w")
tv.heading(3, text="    ",anchor="w")
tv.column("# 1",width=150)
tv.column("# 2",width=110)
tv.column("# 3",width=50)
def update_item():
    selected = tv.focus()
tv.insert('', index=0,values=("Barnhart, Ralph","683-2341"))
tv.insert('', index=1,values=("Jones, Janet", "483-5432"))
tv.insert('', index=2,values=("Meyers, Chris", "343-4349"))
tv.insert('', index=3,values=("Nelson, Eric", "485-2689"))
tv.insert('', index=4,values=("Prefect, Ford","987-6543"))
tv.insert('', index=5,values=("Smith, Bob","689-1234"))
tv.insert('', index=6,values=("Smith, Robert","689-1235"))
tv.insert('', index=7,values=("Shanti, Jayanth","632-1357"))
sb = Scrollbar(frame, orient=VERTICAL)
sb.pack(side=RIGHT, fill=Y)
tv.config(yscrollcommand=sb.set)
sb.config(command=tv.yview)
root.mainloop()
```

OUTPUT:



Name: Barnhart, Ralph

Phone: 683-2341

Name	Phone
Barnhart, Ralph	683-2341
Jones, Janet	483-5432
Meyers, Chris	343-4349
Nelson, Eric	485-2689
Prefect, Ford	987-6543
Smith, Bob	689-1234
Smith, Robert	689-1234

RESULT:

Thus, the given program is executed successfully.

QUESTION 11:

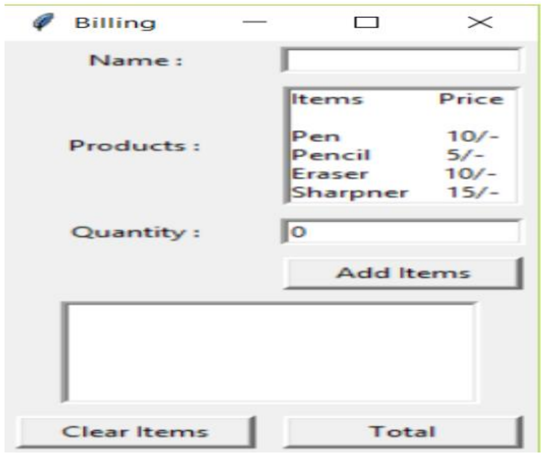
AIM: To create the given design.

CODE:

```
import tkinter as tk
from tkinter import *
root = tk.Tk()
root.geometry("310x300")
root.title('Billing')
l1 = tk.Label(root,text = " Name: ").grid(row=0)
t1=tk.Entry(root,width=30).grid(row=0,column=1)
l2 = tk.Label(root,text = " Products: ").grid(row=2)
t2=tk.Entry(root,width=30).grid(row=2,column=1,padx=10,pady=10,ipady=20)
l3 = tk.Label(root,text = " Quantity: ").grid(row=3)
t3=tk.Entry(root,width=30).grid(row=3,column=1)
w = Button(root,width=25, text = " Add Items ",command = root.destroy).grid(row=4,column=1)
```

```
e=tk.Entry(root,width=16).place(x=15,y=150, width=250,height=100)
w1 = Button(root,width=15, text = " Clear items ",command = root.destroy).place(x=15,y=260)
w2 = Button(root,width=15, text = " Total ",command = root.destroy).place(x=150,y=260)
root.mainloop()
```

OUTPUT:



RESULT:

Thus, the given program is executed successfully.

QUESTION 12:

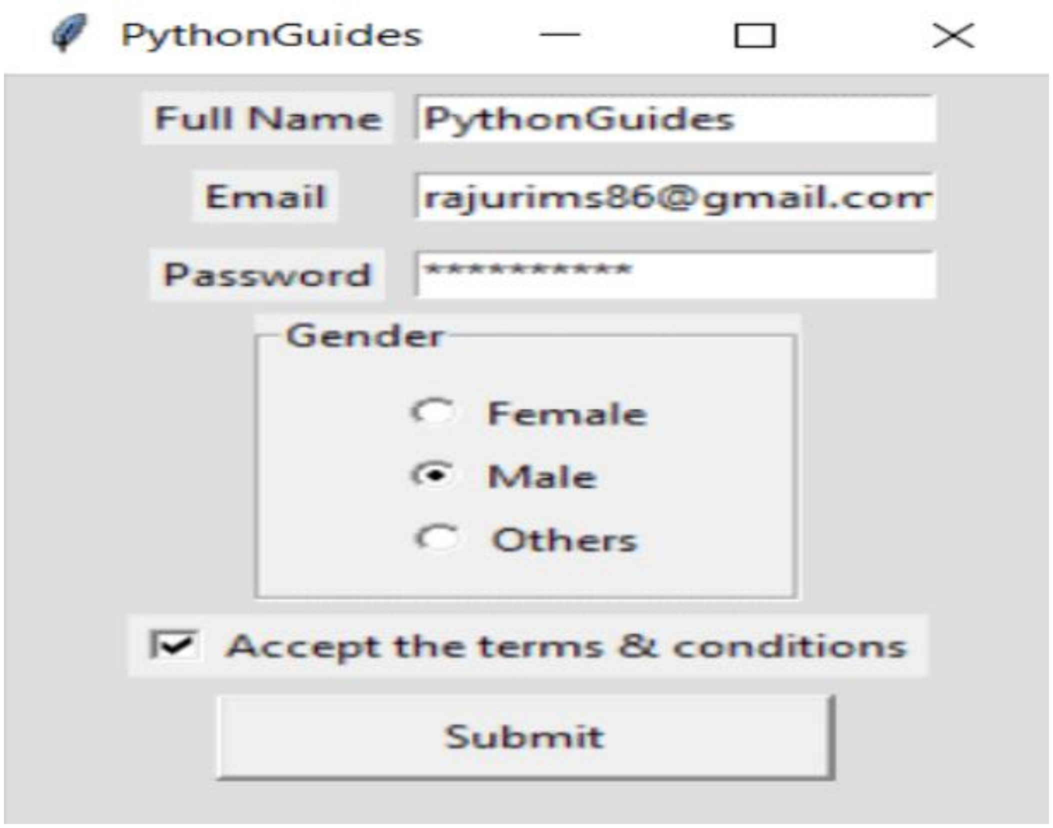
AIM: To create the given design.

CODE:

```
from tkinter import *
root = Tk()
root.geometry("350x310")
root.title('PythonGuides')
root.configure(background="#d1d1c7");
l1 = Label(root,text = "Full Name ",background="#f4f4f3").place(x=50,y=10)
t=Entry(root,width=30).place(x=130,y=10)
l2=Label(root,text="Email ",background="#f4f4f3").place(x=50,y=40)
t=Entry(root,width=30).place(x=130,y=40)
l3=Label(root,text="Password ",background="#f4f4f3").place(x=50,y=65)
t=Entry(root,width=30).place(x=130,y=65)
radio=IntVar()
```

```
frame = Frame(root,background="#f4f4f3",bd=5)

frame.place(x=90,y=100)

w = LabelFrame(frame, text='Gender',background="#f4f4f3", relief=GROOVE,padx=60,pady=15)

w.grid(row=4, column=2)

rb1=Radiobutton(w,text='Female',variable=radio,value=1,background="#f4f4f3").grid(row=5,column=1)

rb2=Radiobutton(w,text='Male',variable=radio,value=2,background="#f4f4f3").grid(row=6,column=1)

rb3=Radiobutton(w,text='Others',variable=radio,value=3,background="#f4f4f3").grid(row=7,column=1)

v1=IntVar()

cb1=Checkbutton(root,text='Accept the terms & conditions',variable=v1,
onvalue=1,background="#f4f4f3",width=35)

cb1.place(x=40,y=240)

button=Button(root, text='Submit', width=25,background="#f4f4f3").place(x=70,y=270)

root.mainloop()
```

OUTPUT:



PythonGuides

Full Name: PythonGuides

Email: rajurims86@gmail.com

Password: **********

Gender

( ) Female

(●) Male

( ) Others

☑ Accept the terms & conditions

RESULT:

Thus, the given program is executed successfully.

QUESTION 13:

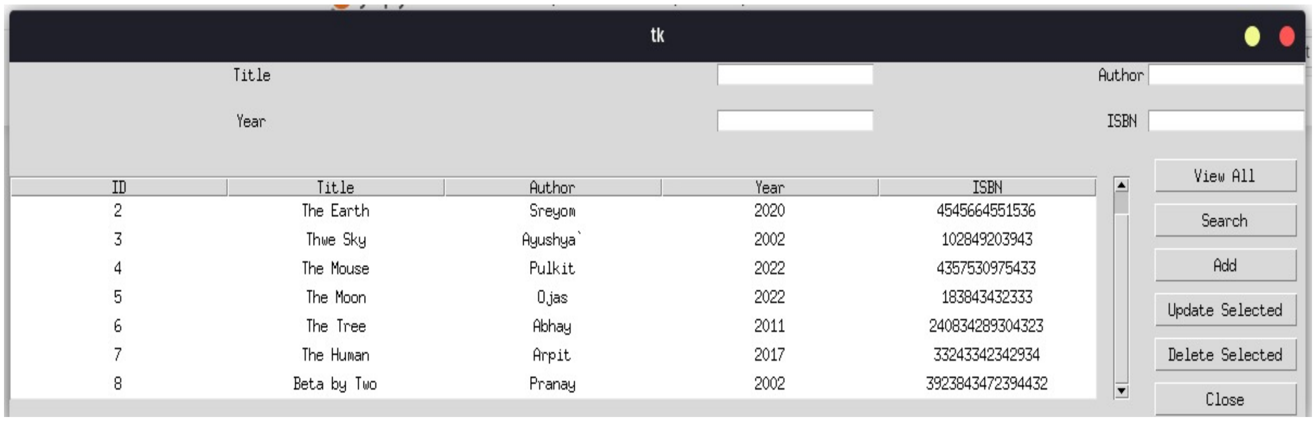
AIM: To create the given design.

CODE:

```
from tkinter import ttk

from tkinter import *

root = Tk()

phone_list = ttk.Style()

tree = ttk.Treeview(root, column=("ID", "Title", "Author", "Year", "ISBN"),

                    show='headings', height=7)

id = 3

tree.column("#1", anchor=CENTER)

tree.heading("# 1", text="ID")

tree.column("# 2", anchor=CENTER)

tree.heading("# 2", text="Title")

tree.column("# 3", anchor=CENTER)

tree.heading("# 3", text="Author")

tree.column("# 4", anchor=CENTER)

tree.heading("# 4", text="Year")

tree.column("# 5", anchor=CENTER)

tree.heading("# 5", text="ISBN")

def add():

    global id

    tree.insert('', 'end', values=(id, title.get(), auth.get(), year.get(),

                                isbn.get()))

    title.delete(0, END)

    auth.delete(0, END)

    year.delete(0, END)

    isbn.delete(0, END)

    id += 1


def delete():

    global id
    selected_item = tree.selection()[0]
    tree.delete(selected_item)
    id -= 1
def edit():
    selected_item = tree.selection()[0]
    tree.item(selected_item, values=(id, title.get(), auth.get(),
                                    year.get(), isbn.get()))
Label(root, text="Title").grid(row=0, column=0)
title = Entry(root)
title.grid(row=0, column=1)
Label(root, text="Author").grid(row=0, column=2)
auth = Entry(root)
auth.grid(row=0, column=3)
Label(root, text="Year").grid(row=1, column=0, pady=15)
year = Entry(root)
year.grid(row=1, column=1, pady=15)
Label(root, text="ISBN").grid(row=1, column=2, pady=15)
isbn = Entry(root)
isbn.grid(row=1, column=3, pady=15)
scroll_bar = ttk.Scrollbar(root,
                          orient="vertical",
                          command=tree.yview)
scroll_bar.grid(row=2, column=2, sticky="nsew", pady=15, rowspan=6)
tree.configure(yscrollcommand=scroll_bar.set)
tree.insert('', 'end', values=('1', 'The Earth', 'Sreyom', '2020',
                              '4545664551536'))
tree.insert('', 'end', values=('2', 'The Moon', 'Ojas', '2022',
                              '5545446546646'))
tree.grid(row=2, columnspan=2, pady=15, rowspan=6)
Button(root, text="View All", command=edit, width=15,
       justify=CENTER).grid(row=2, column=3)
```

```
Button(root, text="Search", command=delete, width=15,
    justify=CENTER).grid(row=3, column=3)
Button(root, text="Add", command=add, width=15, justify=CENTER).grid(row=4,
                                            column=3)
Button(root, text="Update Selected", command=edit, width=15,
    justify=CENTER).grid(row=5, column=3)
Button(root, text="Delete Selected", command=delete, width=15,
    justify=CENTER).grid(row=6, column=3)
Button(root, text="Close", command=root.destroy, width=15,
    justify=CENTER).grid(row=7, column=3)
root.mainloop()
```

OUTPUT:



ID	Title	Author	Year	ISBN
2	The Earth	Sreyom	2020	4545664551536
3	Thwe Sky	Ayushya`	2002	102849203943
4	The Mouse	Pulkit	2022	4357530975433
5	The Moon	Ojas	2022	183843432333
6	The Tree	Abhay	2011	240834289304323
7	The Human	Arpit	2017	33243342342934
8	Beta by Two	Pranay	2002	3923843472394432

RESULT:

Thus, the given program is executed successfully.

QUESTION 14:

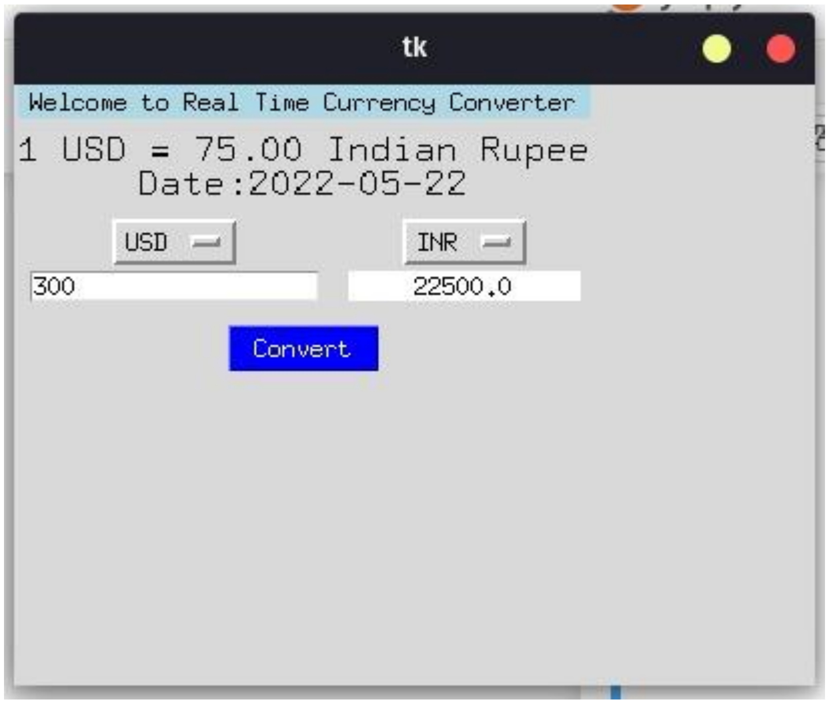
AIM: To create the given design.

CODE:

```
from tkinter import *

root = Tk()

root.geometry("400x300")

title = Label(root, text="Welcome to Real Time Currency Converter",

              width=40, justify=CENTER)

title.config(background="light blue", pady=2, padx=2)

title.grid(row=0, column=0, columnspan=2)

money = Label(root, text="1 USD = 75.00 Indian Rupee\nDate:2022-05-22",

              font=("bold", 13))

money.grid(row=2, column=0, pady=5, columnspan=2)

option = StringVar()

option.set("USD")

option2 = StringVar()

option2.set("INR")

options = ["USD", "INR", "EURO", "POUND"]

currencySelect = OptionMenu(root, option, *options)

currencySelect.grid(row=4, column=0)

moneyEntry = Entry(root, bd=1)

moneyEntry.grid(row=5, column=0)

currencySelect2 = OptionMenu(root, option2, *options)

currencySelect2.grid(row=4, column=1)

moneyEntry2 = Label(root, bg="white", width=16, bd=1)

moneyEntry2.grid(row=5, column=1)



def convert():

    fromCurr = option.get()

    toCurr = option2.get()

    print(fromCurr, toCurr)

    if fromCurr == "USD":
```

```
    if toCurr == "USD":
        moneyEntry2.config(text=str(float(moneyEntry.get()) * 1))
    elif toCurr == "INR":
        moneyEntry2.config(text=str(float(moneyEntry.get()) * 75.00))
    elif toCurr == "EURO":
        moneyEntry2.config(text=str(float(moneyEntry.get())*0.93))
    elif toCurr == "POUND":
        moneyEntry2.config(text=str(float(moneyEntry.get())*0.78))
elif fromCurr == "INR":
    if toCurr == "USD":
        moneyEntry2.config(text=str(float(moneyEntry.get()) * 0.013))
    elif toCurr == "INR":
        moneyEntry2.config(text=str(float(moneyEntry.get()) * 1))
    elif toCurr == "EURO":
        moneyEntry2.config(text=str(float(moneyEntry.get()) * 0.012))
    elif toCurr == "POUND":
        moneyEntry2.config(text=str(float(moneyEntry.get()) * 0.010))
elif fromCurr == "POUND":
    if toCurr == "USD":
        moneyEntry2.config(text=str(float(moneyEntry.get()) * 1.28))
    elif toCurr == "INR":
        moneyEntry2.config(text=str(float(moneyEntry.get()) * 98.19))
    elif toCurr == "EURO":
        moneyEntry2.config(text=str(float(moneyEntry.get()) * 1.19))
    elif toCurr == "POUND":
        moneyEntry2.config(text=str(float(moneyEntry.get()) * 1))
elif fromCurr == "EURO":
    if toCurr == "USD":
        moneyEntry2.config(text=str(float(moneyEntry.get()) * 1.08))
    elif toCurr == "INR":
        moneyEntry2.config(text=str(float(moneyEntry.get()) * 82.74))
        elif toCurr == "EURO":
            moneyEntry2.config(text=str(float(moneyEntry.get()) * 1))
        elif toCurr == "POUND":
            moneyEntry2.config(text=str(float(moneyEntry.get()) * 0.84))



button = Button(root, text="Convert", fg="white", bg="Blue",
                command=convert)
button.grid(row=6, columnspan=2, pady=10)
root.mainloop()
```

OUTPUT:



```
Welcome to Real Time Currency Converter
1 USD = 75.00 Indian Rupee
      Date:2022-05-22
USD    INR
300    22500.0
```

RESULT:

Thus, the given program is executed successfully.

QUESTION 15:

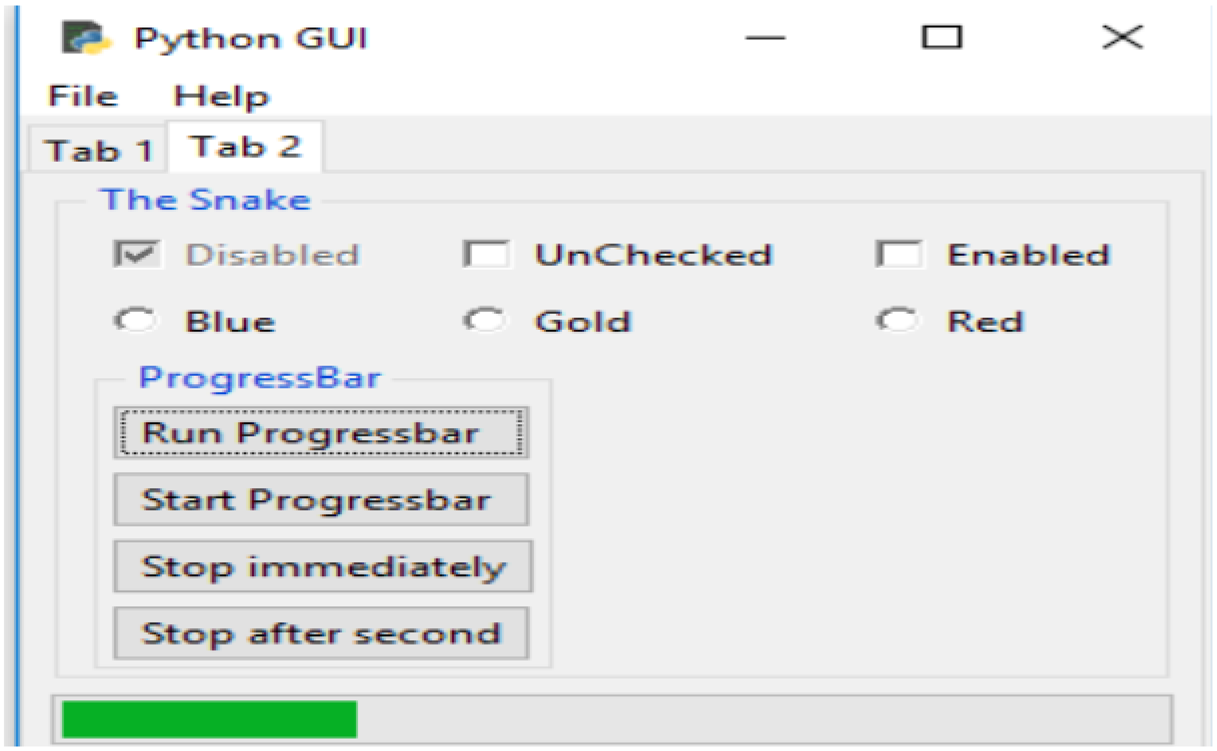
AIM: To create the given design.

CODE:

```
from tkinter import ttk

import tkinter as tk

from tkinter.messagebox import showinfo

root = tk.Tk()

root.geometry('300x120')

root.title('Progressbar')



def update_progress_label():

    return f"Current Progress: {pb['value']}%"



def progress():

    if pb['value'] < 100:

        pb['value'] += 20

        value_label['text'] = update_progress_label()

    else:

        showinfo(message='The progress completed!')



def start():

    for i in range(5):

        progress()



def stop():

    pb.stop()

    value_label['text'] = update_progress_label()
```

```
# progressbar
pb = ttk.Progressbar(
    root,
    orient='horizontal',
    mode='determinate',
    length=280
)
# place the progressbar
pb.grid(column=0, row=0, columnspan=3, padx=10, pady=20)
# label
value_label = ttk.Label(root, text=update_progress_label())
value_label.grid(column=0, row=1, columnspan=3)
# start button
start_button = ttk.Button(
    root,
    text='Progress',
    command=progress
)
start_button.grid(column=0, row=2, padx=10, pady=10, sticky=tk.E)
stop_button = ttk.Button(
    root,
    text='Stop',
    command=stop
)
stop_button.grid(column=1, row=2, padx=10, pady=10, sticky=tk.W)
start_button = ttk.Button(root, text="Start", command=start)
start_button.grid(column=2, row=2, padx=10, pady=10, sticky=tk.E)
root.mainloop()
```

OUTPUT:



RESULT:

Thus, the given program is executed successfully.

EX-6: NETWORK PROGRAMMING

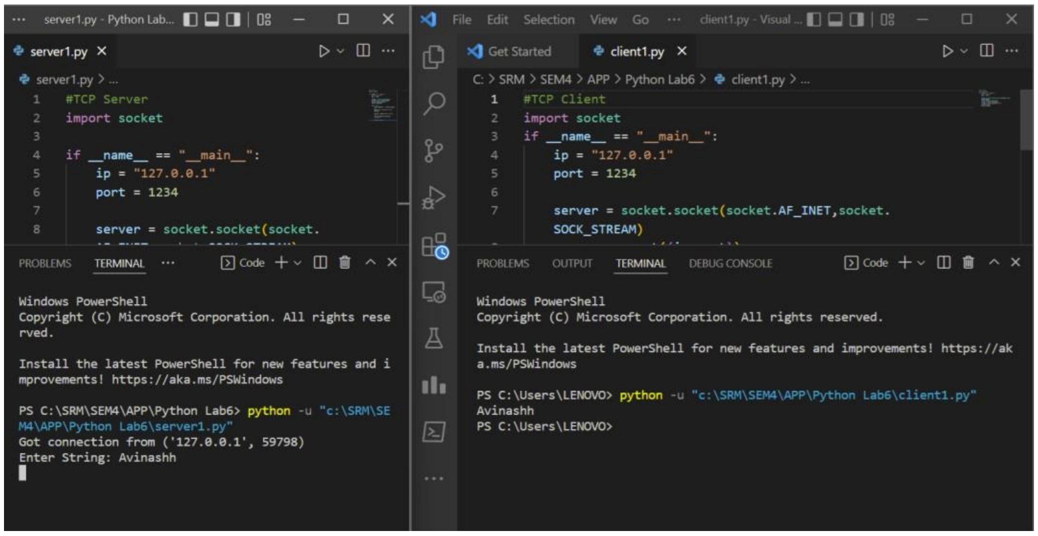
QUESTION 1:

Create Simple Client Server Application using TCP Socket where server issue a command which will be executed at the client slide as a process of remote command execution.

CODE:

```
#TCP Server
import socket
if__name______== "_main
   ": ip = "127.0.0.1"
   port = 1234
   server = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
   server.bind((ip,port))
   server.listen(5)
   while True:
      client,address = server.accept()
      print("Got connection from",address)
      string = input("Enter String: ")
      client.send(bytes(string, "utf-8"))
      client.close()
#TCP Client
import socket
if__name______== "_main
   ": ip = "127.0.0.1"
   port = 1234
   server = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
   server.connect((ip,port))
   string = server.recv(1024)
   string = string.decode("utf-8")
   print(string)
   server.close()
```

OUTPUT:


server1.py - Python Lab...
server1.py
File Edit Selection View Go
client1.py - Visual ...
Get Started
client1.py
C: > SRM > SEM4 > APP > Python Lab6 > client1.py > ...
PROBLEMS
OUTPUT
TERMINAL
DEBUG CONSOLE
Code

```
#TCP Server
import socket

if __name__ == "__main__":
    ip = "127.0.0.1"
    port = 1234

    server = socket.socket(socket.
```

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights rese
rved.

Install the latest PowerShell for new features and i
mprovements! https://aka.ms/PSWindows

PS C:\SRM\SEM4\APP\Python Lab6> python -u "c:\SRM\SE
M4\APP\Python Lab6\server1.py"
Got connection from ('127.0.0.1', 59798)
Enter String: Avinashh
```

```
#TCP Client
import socket
if __name__ == "__main__":
    ip = "127.0.0.1"
    port = 1234

    server = socket.socket(socket.AF_INET,socket.
    SOCK_STREAM)
```

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://ak
a.ms/PSWindows

PS C:\Users\LENOVO> python -u "c:\SRM\SEM4\APP\Python Lab6\client1.py"
Avinashh
PS C:\Users\LENOVO>
```

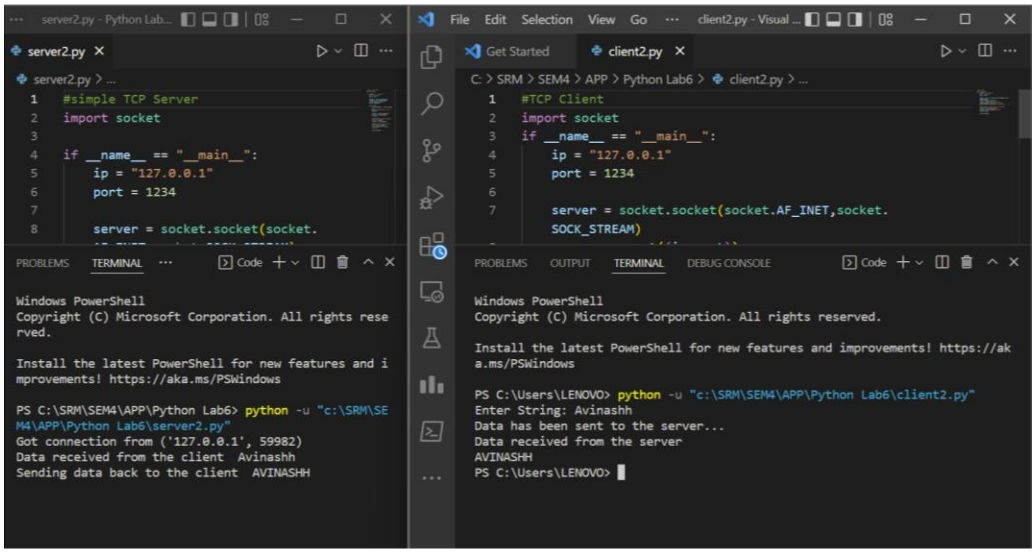
QUESTION 2:

Write a Socket-based Python server program that responds to client messages as follows: When it receives a message from a client, it simply converts the message into all uppercase letters and sends back the same to the client. Write both client and server programs demonstrating this.

CODE:

```
#simple TCP Server
import socket
if___name______== "_main
    ": ip = "127.0.0.1"
    port = 1234
    server = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
    server.bind((ip,port))
    server.listen(5)
    while True:
        client,address = server.accept()
        print("Got connection from",address)
        string = client.recv(1024)
        string = string.decode("utf-8")
        print("Data received from the client ",string)
        string = string.upper()
        print("Sending data back to the client ",string)
        client.send(bytes(string, "utf-8"))
        client.close()
#TCP Client
import socket
if__name______== "_main
    ": ip = "127.0.0.1"
    port = 1234
    server = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
    server.connect((ip,port))
    string = input("Enter String: ")
    server.send(bytes(string, "utf-8"))
    print("Data has been sent to the server... ")
    print("Data received from the server")
    string = server.recv(1024)
    string = string.decode("utf-8")
    print(string)
    server.close()
```

OUTPUT:



```python
#simple TCP Server
import socket

if __name__ == "__main__":
    ip = "127.0.0.1"
    port = 1234

    server = socket.socket(socket.
```

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights rese
rved.

Install the latest PowerShell for new features and i
mprovements! https://aka.ms/PSWindows

PS C:\SRM\SEM4\APP\Python Lab6> python -u "c:\SRM\SE
M4\APP\Python Lab6\server2.py"
Got connection from ('127.0.0.1', 59982)
Data received from the client  Avinashh
Sending data back to the client  AVINASHH
```

```python
#TCP Client
import socket
if __name__ == "__main__":
    ip = "127.0.0.1"
    port = 1234

    server = socket.socket(socket.AF_INET,socket.
    SOCK_STREAM)
```

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://ak
a.ms/PSWindows

PS C:\Users\LENOVO> python -u "c:\SRM\SEM4\APP\Python Lab6\client2.py"
Enter String: Avinashh
Data has been sent to the server...
Data received from the server
AVINASHH
PS C:\Users\LENOVO>
```

EX-7: CONCURRENT PROGRAMMING

QUESTION 1:

In computing, the producer-consumer problem (also known as the bounded-buffer problem) is a classic example of a multi-process synchronization problem. The problem describes two processes, the producer and the consumer, which share a common, fixed-size buffer used as a queue.

● The producer's job is to generate data, put it into the buffer, and start again.

● At the same time, the consumer is consuming the data (i.e. removing it from the buffer), one piece at a time.

Problem

To make sure that the producer won't try to add data into the buffer if it's full and that the consumer won't try to remove data from an empty buffer.

Solution

The producer is to either go to sleep or discard data if the buffer is full. The next time the consumer removes an item from the buffer, it notifies the producer, who starts to fill the buffer again. In the same way, the consumer can go to sleep if it finds the buffer to be empty. The next time the producer puts data into the buffer, it wakes up the sleeping consumer.

CODE:

```
import threading
import time
items=[]
items_cv=threading.Condition()
def prod():
    print("I'm the producer")
    for i in range(30):
        with items_cv:
            items.append(i)
            items_cv.notify()
        time.sleep(1)
def consumer():
    print("I'm a consumer",threading.current_thread().name)
    while True:
        with items_cv:
            while not items:
                items_cv.wait()
            x = items.pop(0)
        print(threading.current_thread().name,"got",x)
        time.sleep(5)
con=[threading.Thread(target=consumer)
        for i in range(10)]
for co in con:
    co.daemon=True
    co.start()
prod()
```

OUTPUT:

```
I'm a consumer I'm a consumerThread-1 (consumer)
 I'm a consumer Thread-2 (consumer)
I'm a consumerI'm a consumerThread-3 (consumer)
  Thread-5 (consumer)I'm a consumerI'm a consumer Thread-4 (consumer)Thread-7 (consumer)
I'm a consumer


Thread-10 (consumer) got 9
Thread-1 (consumer) got 10
Thread-2 (consumer) got 11
Thread-3 (consumer) got 12
Thread-5 (consumer) got 13
Thread-7 (consumer) got 14
Thread-4 (consumer) got 15
Thread-6 (consumer) got 16
Thread-8 (consumer) got 17
```

```
Thread-9 (consumer) got 18
Thread-10 (consumer) got 19
Thread-1 (consumer) got 20
Thread-2 (consumer) got 21
Thread-3 (consumer) got 22
Thread-5 (consumer) got 23
Thread-7 (consumer) got 24
Thread-4 (consumer) got 25
Thread-6 (consumer) got 26
Thread-8 (consumer) got 27
Thread-9 (consumer) got 28
Thread-10 (consumer) got 29
```

RESULT:

Thus, the given program is executed successfully.

QUESTION 2:

Problem Statement – We have a buffer of fixed size. A producer can produce an item and can place in the buffer. A consumer can pick items and can consume them. We need to ensure that when a producer is placing an item in the buffer, then at the same time consumer should not consume any item. In this problem, buffer is the critical section.
To solve this problem, we need two counting semaphores – Full and Empty. “Full” keeps track of number of items in the buffer at any given time and “Empty” keeps track of number of unoccupied slots.

CODE:

```
import threading
import time
Cap=10
buf=[-1 for i in range(Cap)]
inind=0
outind=0
```

```
mutex=threading.Semaphore()
empty=threading.Semaphore(Cap)
full=threading.Semaphore(0)
class Producer(threading.Thread):
  def run(self):
    global Cap, buf, inind, outind
    global mutex, empty, full
    items_produced = 0
    counter = 0
    while items_produced < 20:
      empty.acquire()
      mutex.acquire()
      counter += 1
      buf[inind] = counter
      inind = (inind + 1)%Cap
      print(f"Producer produced:{counter}")
      mutex.release()
      full.release()
      time.sleep(1)
      items_produced += 1
class Consumer(threading.Thread):
  def run(self):
    global Cap, buf, inind, outind, counter
    global mutex, empty, full
    items_consumed = 0
    while items_consumed < 20:
      full.acquire()
      mutex.acquire()
      item = buf[outind]
      outind = (outind + 1)%Cap
    print(f"Consumer consumed item:{item}")
    mutex.release()
    empty.release()
    time.sleep(2.5)
    items_consumed += 1
producer=Producer()
consumer=Consumer()
consumer.start()
producer.start()
producer.join()
consumer.join()
```

OUTPUT:

```
Producer produced:1
Consumer consumed item:1
Producer produced:2
Producer produced:3
Consumer consumed item:2
Producer produced:4
Producer produced:5
Consumer consumed item:3
Producer produced:6
Producer produced:7
Producer produced:8
Consumer consumed item:4
Producer produced:9
Producer produced:10
Consumer consumed item:5
Producer produced:11
Producer produced:12
Producer produced:13
```

```
Consumer consumed item:6
Producer produced:14
Producer produced:15
Consumer consumed item:7
Producer produced:16
Producer produced:17
Consumer consumed item:8
Producer produced:18
Consumer consumed item:9
Producer produced:19
Consumer consumed item:10
Producer produced:20
Consumer consumed item:11
Consumer consumed item:12
Consumer consumed item:13
Consumer consumed item:14
Consumer consumed item:15
Consumer consumed item:16
Consumer consumed item:17
Consumer consumed item:18
Consumer consumed item:19
Consumer consumed item:20
```

RESULT:

Thus, the given program is executed successfully.

EX-8: PARALLEL PROGRAMMING

QUESTION 1:

The Dining Philosopher Problem – The Dining Philosopher Problem states that K philosophers seated around a circular table with one chopstick between each pair of philosophers. There is one chopstick between each philosopher. A philosopher may eat if he can pick up the two chopsticks adjacent to him. One chopstick may be picked up by any one of its adjacent followers but not both.

There are three states of philosopher: THINKING, HUNGRY and EATING. Here there are two semaphores: Mutex and a semaphore array for the philosophers. Mutex is used such that no two philosophers may access the pickup or putdown at the same time. The array is used to control the behavior of each philosopher. But semaphores can result in deadlock due to programming errors.

CODE:

```
import threading
import random
import time
class Philosopher(threading.Thread):
    running = True
    def_init_(self, index, forkOnLeft, forkOnRight):
        threading.Thread._init_(self)
        self.index = index
        self.forkOnLeft = forkOnLeft
        self.forkOnRight = forkOnRight
    def run(self):
        while(self.running):
            time.sleep(30)
            print (f'Philosopher {self.index} is hungry.')
            self.dine()
    def dine(self):
        fork1, fork2 = self.forkOnLeft, self.forkOnRight
        while self.running:
            fork1.acquire()
            locked = fork2.acquire(False)
            if locked: break
            fork1.release()
            print (f'Philosopher {self.index} swaps forks.')
            fork1, fork2 = fork2, fork1
        else:
            return
        self.dining()
        fork2.release()
        fork1.release()
    def dining(self):
        print (f'Philosopher {self.index} starts eating.')
        time.sleep(30)
        print (f'Philosopher {self.index} finishes eating and leaves to think.')
def main():
    forks = [threading.Semaphore() for n in range(5)]
    philosophers= [Philosopher(i, forks[i%5], forks[(i+1)%5])
            for i in range(5)]
    Philosopher.running = True
    for p in philosophers: p.start()
    time.sleep(100)
    Philosopher.running = False
    print ("Now we are finishing.")
if__name______== "
    main_": main()
```

OUTPUT:

```
Philosopher 3 is hungry.Philosopher 0 is hungry.Philosopher 2 is hungry.
Philosopher 1 is hungry.
Philosopher 4 is hungry.


Philosopher 1 swaps forks.
Philosopher 2 starts eating.Philosopher 4 swaps forks.


Philosopher 0 starts eating.
Philosopher 2 finishes eating and leaves to think.
Philosopher 0 finishes eating and leaves to think.Philosopher 3 starts eating.
Philosopher 1 swaps forks.
Philosopher 4 swaps forks.


Philosopher 1 starts eating.
Philosopher 0 is hungry.
Philosopher 3 finishes eating and leaves to think.Philosopher 2 is hungry.Philosopher 1 finishes eating and leaves to think.
Philosopher 0 swaps forks.


Philosopher 4 starts eating.
Philosopher 2 starts eating.
Philosopher 0 swaps forks.


Now we are finishing.
Philosopher 1 is hungry.
Philosopher 3 is hungry.
Philosopher 4 finishes eating and leaves to think.Philosopher 2 finishes eating and leaves to think.


Philosopher 0 starts eating.
Philosopher 0 finishes eating and leaves to think.
```

RESULT:

Thus, the given program is executed successfully.

QUESTION 2:

Consider a situation where we have a file shared between many people.

● If one of the people tries editing the file, no other person should be reading or writing at the same time, otherwise changes will not be visible to him/her.

● However if some person is reading the file, then others may read it at the same time. Precisely in OS we call this situation as the readers-writers problem

Problem parameters:

● One set of data is shared among a number of processes

● Once a writer is ready, it performs its write. Only one writer may write at a time

● If a process is writing, no other process can read it

● If at least one reader is reading, no other process can write

● Readers may not write and only read

CODE:

```
import threading as thread

import random

global x

x = 0

lock = thread.Lock()

def Reader():

    global x

    print('\nReader is Reading!')

    lock.acquire()

    print(f'\nShared Data:{x}')

    lock.release()

    print()

def Writer():

    global x

    print('\nWriter is Writing!')

    lock.acquire()
    x += 1
    print('Writer is Releasing the lock!')
    lock.release()
    print()
if__name______== '
    main_': for i in
    range(0, 10):
        randomNumber = random.randint(0, 100)
        if(randomNumber > 50):
            t1 = thread.Thread(target = Reader)
            t1.start()
        else:
            t2 = thread.Thread(target = Writer)
            t2.start()
t1.join()
t2.join()
```

OUTPUT:

```
Reader is Reading!


Shared Data:0
Reader is Reading!


Writer is Writing!


Writer is Releasing the lock!


Reader is Reading!


Reader is Reading!
```

```
Writer is Writing!

Reader is Reading!

Shared Data:1

Writer is Writing!

Reader is Reading!

Reader is Reading!




Shared Data:1



Shared Data:1



Shared Data:1

Writer is Releasing the lock!


Writer is Releasing the lock!




Shared Data:3



Shared Data:3
```

RESULT:

Thus, the given program is executed successfully.

EX-9: FUNCTIONAL PROGRAMMING

QUESTION 1:

Write a Python program to create Fibonacci series up to n using Lambda.

CODE:

```
def fibonacci(count):
   listA=[0,1]
   any(map(lambda _:listA.append(sum(listA[-2:])),range(2, count)))
   return listA[:count]
n=int(input())
print(fibonacci(n))
```

OUTPUT:

```
5
[0, 1, 1, 2, 3]
```

RESULT:

Thus, the given program is executed successfully.

QUESTION 2:

Write a Python program to find the numbers of a given string and store them in a list, display the numbers which are bigger than the length of the list in sorted form. Use lambda function to solve the problem.

CODE:

```
str1 = "SOC 23 CTech 5 DSBS8 NWC 56 CINtel 20 5"
print("Original string:",str1)
str_num=[i for i in str1.split(' ')]
lenght=len(str_num)
numbers=sorted([int(x) for x in str_num if x.isdigit()])
print('Numbers in sorted form:')
for i in ((filter(lambda x:x>lenght,numbers))):
    print(i,end=' ')
```

OUTPUT:

```
Original string: SOC 23 CTech 5 DSBS8 NWC 56 CINtel 20 5
Numbers in sorted form:
20 23 56
```

RESULT:

Thus, the given program is executed successfully.

QUESTION 3:

Write a Python program to sort a given list of strings(numbers) numerically using lambda.

CODE:

```
def sort_numeric_strings(nums_str):
    result = sorted(nums_str, key=lambda el: int(el))
    return result
nums_str=['40','20','50','47','30','-10','-20','115','100']
print("Original list:")
print(nums_str)
print("Sort the said list of strings(numbers) numerically:")
print(sort_numeric_strings(nums_str))
```

OUTPUT:

```
Original list:
['40', '20', '50', '47', '30', '-10', '-20', '115', '100']
Sort the said list of strings(numbers) numerically:
['-20', '-10', '20', '30', '40', '47', '50', '100', '115']
```

RESULT:

Thus, the given program is executed successfully.

QUESTION 4:

Write a Python program to calculate the average value of the numbers in a given tuple of tuples using lambda.

CODE:

```
def average_tuple(nums):
    result = tuple(map(lambda x: sum(x) / float(len(x)), zip(*nums)))
    return result
nums = ((1,1,1),(3,4,5),(8,7,9),(10,20,30))
print("Original Tuple:")
print(nums)
print("Average value of the numbers of the said tuple of tuples:\n",average_tuple(nums))
```

OUTPUT:

```
Original Tuple:
((1, 1, 1), (3, 4, 5), (8, 7, 9), (10, 20, 30))
Average value of the numbers of the said tuple of tuples:
 (5.5, 8.0, 11.25)
```

RESULT:

Thus, the given program is executed successfully.

QUESTION 5:

Write a Python program to find the nested lists elements, which are present in another list using lambda.

CODE:

```
def intersection_nested_lists(l1, l2):
    result = [list(filter(lambda x: x in l1, sublist)) for sublist in l2]
    return result
nums1 = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]
nums2 = [[12, 18, 23, 25, 45], [7, 11, 19, 24, 28], [1, 5, 8, 18, 15, 16]]
print("Original lists:")
print(nums1)
print(nums2)
print("Intersection of said nested lists:")
print(intersection_nested_lists(nums1, nums2))
```

OUTPUT:

```
Original lists:
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]
[[12, 18, 23, 25, 45], [7, 11, 19, 24, 28], [1, 5, 8, 18, 15, 16]]
Intersection of said nested lists:
[[12], [7, 11], [1, 5, 8]]
```

RESULT:

Thus, the given program is executed successfully.

QUESTION 6:

Write a Python program to convert a given list of strings into list of lists using map function.

CODE:

```
def strings_to_listOflists(str):
    result=map(list, str)
    return list(result)
colors=["Red","Blue","Orange"]
print('Original list of strings:')
print(colors)
print("Convert the said list of strings into list of lists:")
print(strings_to_listOflists(colors))
```

OUTPUT:

```
Original list of strings:
['Red', 'Blue', 'Orange']
Convert the said list of strings into list of lists:
[['R', 'e', 'd'], ['B', 'l', 'u', 'e'], ['O', 'r', 'a', 'n', 'g', 'e']]
```

RESULT:

Thus, the given program is executed successfully.

QUESTION 7:

Write a Python program to convert all the characters in uppercase and lowercase and eliminate duplicate letters from a given sequence. Use map() function.

CODE:

```
def change_cases(s):
  return str(s).upper(), str(s).lower()
chrars={'a','b','C','d','a','e','f','G','a'}
print("Original Characters:\n",chrars)
result=map(change_cases, chrars)
print("After converting above characters in upper and lower cases\nand eliminating duplicate letters:")
print(set(result))
```

OUTPUT:

```
Original Characters:
 {'C', 'a', 'e', 'G', 'd', 'b', 'f'}
After converting above characters in upper and lower cases
and eliminating duplicate letters:
{('B', 'b'), ('C', 'c'), ('F', 'f'), ('G', 'g'), ('E', 'e'), ('A', 'a'), ('D', 'd')}
```

RESULT:

Thus, the given program is executed successfully.

QUESTION 8:

Write a Python program to add two given lists and find the difference between lists.

Use map() function.

CODE:

```
def addition_subtrction(x, y):
    return x + y, x - y
nums1=[80,70,60,50]
nums2=[40,30,20,10]
print("Original lists:")
print(nums1)
print(nums2)
result=map(addition_subtrction, nums1, nums2)
print("Result:")
print(list(result))
```

OUTPUT:

```
Original lists:
[80, 70, 60, 50]
[40, 30, 20, 10]
Result:
[(120, 40), (100, 40), (80, 40), (60, 40)]
```

RESULT:

Thus, the given program is executed successfully.

QUESTION 9:

Write a Python program to add two given lists and find the difference between lists.

Use map() function.

CODE:

```
def addition_subtrction(x, y):
    return x + y, x - y
nums1=[80,70,60,50]
nums2=[40,30,20,10]
print("Original lists:")
print(nums1)
print(nums2)
result=map(addition_subtrction, nums1, nums2)
print("Result:")
print(list(result))
```

OUTPUT:

```
Original lists:
[80, 70, 60, 50]
```

```
[40, 30, 20, 10]
Result:
[(120, 40), (100, 40), (80, 40), (60, 40)]
```

RESULT:

Thus, the given program is executed successfully.

QUESTION 10:

Filter the array, and return a new array with only the values equal to or above 18 (consider filter function)

CODE:

```
def fun(variable):
    letters = ['a', 'e', 'i', 'o', 'u']
    if (variable in letters):
        return True
    else:
        return False
sequence = ['g', 'e', 'e', 'j', 'k', 's', 'p', 'r']
filtered=filter(fun, sequence)
print('The filtered letters are:')
for s in filtered:
    print(s)
```

OUTPUT:

```
The filtered letters are:
e
e
```

RESULT:

Thus, the given program is executed successfully.

QUESTION 11:

Write a Python program to filter only vowels from given sequence.

CODE:

```
def vowel(c):
    newstr = c
    vowels = ['a', 'e', 'i', 'o', 'u']
    for x in c.lower():
        if x not in vowels:
            newstr = newstr.replace(x,"")
    return newstr
seq='geejkspr'
print(vowel(seq))
```

OUTPUT:

```
ee
```

RESULT:

Thus, the given program is executed successfully.

QUESTION 12:

Write a Python program to calculate sum of numbers from the list and maximum element from the list (use reduce function)

CODE:

```
from functools import reduce
def sum(a, b):
    print(f"a={a},b={b},{a}+{b}={a+b}")
    return a + b
scores = [75, 65, 80, 95, 50]
total = reduce(sum, scores)
print(total)
```

OUTPUT:

```
a=75,b=65,75+65=140
a=140,b=80,140+80=220
a=220,b=95,220+95=315
```

```
a=315,b=50,315+50=365
365
```

RESULT:

Thus, the given program is executed successfully.

QUESTION 13:

Write a python program to calculate factorial of given number (use reduce function)

CODE:

```
import functools
def mult(x,y):
    print(f"x={x} y={y}")
    return x*y
fact=functools.reduce(mult, range(1, 4))
print(f'Factorial of 3:{fact}')
```

OUTPUT:

```
x=1 y=2
x=2 y=3
Factorial of 3:6
```

RESULT:

Thus, the given program is executed successfully.

EX-10: SYMBOLIC PROGRAMMING

QUESTION 1:

Solve the following using symbolic paradigm:

i. Calculate sqrt (2) with 100 decimals.

CODE:

```
from sympy import *
print(N(sqrt(2),100))
```

OUTPUT:

```
1.414213562373095048801688724209698078569671875376948073176679737990732478462107038850387534327641573
```

ii. Calculate (1/2+1/3) in rational arithmetic.

CODE:

```
import sympy as sym
a = (sym.Rational(1, 2)+sym.Rational(1,3))
print(sym.simplify(a))
```

OUTPUT:

```
5/6
```

iii. Calculate the expanded form of (x+y) ^ 6.

CODE:

```
import sympy as sym
x = sym.Symbol('x')
y = sym.Symbol('y')
a = sym.expand((x+y)**6)
print(a)
```

OUTPUT:

```
x**6 + 6*x**5*y + 15*x**4*y**2 + 20*x**3*y**3 + 15*x**2*y**4 + 6*x*y**5 + y**6
```

iv. Simplify the trigonometric expression sin (x) / cos (x)

CODE:

```
from sympy import *
x = symbols('x')

a = (sin(x)/(cos(x)))
s=trigsimp(a)
print(format(s))
```

OUTPUT:

tan(x)

v. Calculate sin x -xx^3n

CODE:

```
from sympy import *
x,n=symbols('x n')
exp= sin(x)-x*x**3*n
s=trigsimp(exp)
print(s)
```

OUTPUT:

```
-n*x**4 + sin(x)
```

RESULT:

Thus, the given program is executed successfully.

QUESTION 2:

Develop a python code for to carry out the operations on the given algebraic manipulation for

the given expression a2 −ab+ab−b2=a2−b2 by using the symbolic programming paradigms principles.

CODE:

```
import sympy as sym
a = sym.Symbol('a')
b = sym.Symbol('b')
lhs=sym.simplify(a**2-a*b+a*b-b**2)
rhs=sym.simplify(a**2-b**2)
print('lhs is ',lhs)
print('rhs is ',rhs)
```

OUTPUT:

```
lhsis a**2 - b**2

rhsis a**2 - b**2
```

RESULT:

Thus, the given program is executed successfully.

QUESTION 3:

Give the Symbolic program for the expression given below:

a. ∬ a 2 da

CODE:

```
from sympy import *
import functools
import operator
import random
a=symbols('a')
g=a**2
j=integrate(g,a)
print('before integration a**2')
print('after integration '+str(j))
```

OUTPUT:

```
before integration a**2

after integration a**3/3
```

b. 2x+y 2

CODE:

```
import sympy as sym
x = sym.Symbol('x')
y = sym.Symbol('y')
a = sym.simplify(2*x+y**2)
print(a)
```

OUTPUT:

```
2*x + y**2
```

c. 1/10 + 1/5

CODE:

```
import sympy as sym
a = (sym.Rational(1, 10)+sym.Rational(1,5))
print(sym.simplify(a))
```

OUTPUT:

```
3/10
```

d. d/dx(sin(x))

CODE:

```
from sympy import *
import functools
import operator
import random
a=symbols('a')
x=symbols('x')
g=sin(x)
j=diff(g,a)
```

```
print('before differentiation sin(x)')
print('after differentiation '+str(j))
```

OUTPUT:

```
before differentiation sin(x)
after differentiation 0
```

RESULT:

Thus, the given program is executed successfully.

EX-11: LOGICAL PROGRAMMING

QUESTION 1:

Implement using pyDatalog:

Assume given a set of facts of the form father(name1,name2) (name1 is the father of name2).

a. Define a predicate brother(X,Y) which holds iff X and Y are brothers.

b. Define a predicate cousin(X,Y) which holds iff X and Y are cousins.

c. Define a predicate grandson(X,Y) which holds iff X is a grandson of Y.

d. Define a predicate descendent(X,Y) which holds iff X is a descendent of Y.

e. Consider the following genealogical

```
        tree: a
       / \
      b c
     / \ |
    d e f
```

What are the answers generated by your definitions for the queries:

brother(X,Y)

cousin(X,Y)

grandson(X,Y)

descendent(X,Y)

CODE:

```
from pyDatalog import pyDatalog

pyDatalog.create_terms('a,b,c,d,e,f,brother,cousin,grandson,descendent,X,Y')

+brother('b','c')

+brother('d','e')

+cousin('d','f')

+cousin('e','f')

+grandson('d','a')

+grandson('e','a')

+grandson('f','a')
```

```
+descendent('b','a')
+descendent('c','a')
+descendent('d','b')
+descendent('f','c')
print(pyDatalog.ask('brother(X,Y)'))
print(pyDatalog.ask('cousin(X,Y)'))
print(pyDatalog.ask('grandson(X,Y)'))
print(pyDatalog.ask('descendent(X,Y)'))
```

OUTPUT:

```
{('d', 'e'), ('b', 'c')}
{('d', 'f'), ('e', 'f')}
{('e', 'a'), ('d', 'a'), ('f', 'a')}
{('f', 'c'), ('b', 'a'), ('c', 'a'), ('d', 'b')}
```

RESULT:

Thus, the given program is executed successfully.

QUESTION 2:

Encode the following facts and rules in pyDatalog:

- Bear is big
- Elephant is big
- Cat is small
- Bear is brown
- Cat is black
- Elephant is gray
- An animal is dark if it is black
- An animal is dark if it is brown

Write a query to find which animal is dark and big.

CODE:

```
pyDatalog.create_terms('X,Y,Z,bear,elephant,cat,small,big,brown,black,gray,dark')
+big('elephant')
```

```
+big('bear')
+small('cat')
+black('cat')
+brown('bear')
+gray('elephant')
dark(X)<=black(X) or brown(X)
print(big(X),dark(X))
```

OUTPUT:

```
X
-----------
bear
elephant X
---
cat
```

RESULT:

Thus, the given program is executed successfully.

QUESTION 3:

The following are the marks scored by 5 students.

Student Name	Mark
Ram	90
Raju	45
Priya	85
Carol	70
Shyam	80

Enter the above data using pyDatalog.

Write queries for the following:

a. Print Student name and mark of all students.

b. Who has scored 80 marks?

c. What mark has been scored by Priya?

d. Write a rule ‘passm’ denoting that pass mark is greater than 50. Use the rule to print all students who failed.

e. Write rules for finding grade letters for a marks and use the rule to find the grade letter of a given mark.

CODE:

```
from pyDatalog import pyDatalog
pyDatalog.create_terms('X,Y,Z,student,marks,passm,grades')
+student('ram')
+student('raju')
+student('priya')
+student('carol')
+student('shyam')
+marks('90','ram')
+marks('45','raju')
+marks('85','priya')
+marks('70','carol')
+marks('80','shyam')
+grades('ram','O')
+grades('priya','A')
+grades('shyam','A')
+grades('carol','B')
+grades('raju','E')
print(marks(X,Y))
print(marks('80',X))
print(marks(X,'priya'))
passm(X)<=grades(X,'E')
print(passm(X))
```

OUTPUT:

```
X | Y
----|-----------
```

```
80 | shyam
70 | carol
85 | priya
45 | raju
90 | ram
X
-------
shyam
X
--
85
X
- - -
raju
```

RESULT:

Thus, the given program is executed successfully.

QUESTION 4:

Solve the set of queries in the previous question using imperative programming paradigm in Python. Store the data in a dictionary.

CODE:

```
marks={90:'ram', 85: 'priya', 80: 'shyam', 70: 'carol',45:'raju'}
for i in marks:
    print(i,marks[i])
print(marks [80])
for i in marks:
    if marks[i]=='priya':
        print(i)
for i in marks:
    if i<50:
        print(marks[i])
for i in marks:
    if i>=90:
        print (marks[i], '0')
    elif i<90 and i>=80:
        print (marks[i], 'A')
    elif i<80 and i>=70:
        print(marks[i], 'B')
    elif i<70 and i>=60:
        print (marks[i], 'C')
    elif i<60 and i>=50:
        print (marks[i], 'D')
    else:
        print(marks[i], 'E')
```

OUTPUT:

```
90 ram
85 priya
80 shyam
70 carol
45 raju
shyam
85
raju
ram 0
priya A
shyam A
carol B
raju E
```

RESULT:

Thus, the given program is executed successfully.

QUESTION 5:

Write a recursive program to find factorial of a number using pyDatalog.

CODE:

```
from pyDatalog import pyDatalog

pyDatalog.create_terms('factorial, N')

num=int(input('Enter any number:'))

factorial[N] = N*factorial[N-1]

factorial[1] = 1

print(factorial[num]==N)
```

OUTPUT:

```
Enter any number:3

N

-

6
```

RESULT:

Thus, the given program is executed successfully.

FUNCTIONAL PROGRAMMING

QUESTION 1:

Calculate the following using Lambda calculus:

a. T AND F

b. 3 * 4

CODE:

```
y=lambda s: s and False
print("T AND F = ",y(True))
x = lambda a : a*4
print("3*4= ",x(3))
```

OUTPUT:

```
T AND F = False
3*4= 12
```

RESULT:

Thus, the given program is executed successfully.

QUESTION 2:

Lambda functions

a. Write a lambda function to convert measurements from meters to feet.

CODE:

```
x=lambda x:3.281*x
print("Answer is: ",x(91))
```

OUTPUT:

```
Answer is:  298.571
```

RESULT:

Thus, the given program is executed successfully.

QUESTION 3:

Passing and returning a function as an argument

Define a function 'square' for squaring a number. Define a function named 'twice' that takes a function f as an argument and returns f(f(x)). Using 'twice' and 'square' create a function 'quad' that takes n as an argument and returns n4 . 'quad' should not be defined explicitly. It should only be created as a variable which is then assigned a function.

CODE:

```
def square(number):
    value=number*number
    return value

   def twice(func,num):
    return func(func(num))
a=int(input('enter a number:'))
b=twice(square,a)
print(b)
```

OUTPUT:

```
enter a number:9
6561
```

RESULT:

Thus, the given program is executed successfully.

QUESTION 4:

Closure

A Closure is a function object that remembers values in enclosing scopes even if they are not present in memory. We have a closure in Python when a nested function references a value in its enclosing scope.

CODE:

```
def smart(fun):

    def inner(a,b):

        if a<b:

a,b=b,a

        return fun(a,b)

    return inner

@smart

def div(a,b):

    print(a/b)

div(2,4)
```

OUTPUT:

```
2.0
```

RESULT:

Thus, the given program is executed successfully.

QUESTION 5:

a. Study the following program by executing it:

CODE:

```
def multiplier_of(n):

def multiplier(number):

return number*n

return multiplier

multiplywith5 = multiplier_of(5)

print(multiplywith5(9))
```

OUTPUT:

```
45
```

b. In a lottery system, random number is chosen by retrieving the number from a random index from a list of random numbers. Write a program to choose a random number in this way. You must use nested functions – the inner function chooses a number from a random index and the outer function generates a random list of numbers. The outer function takes n as a parameter where is the maximum number that can be put in the random list. (Your code should be similar to the program in 5a)

CODE:

```
import random

m=10
print("Let the maximum Limit for generating random numbers be "+str(m))
lst=[]
for i in range(m):

lst.append(random.randint(1,m))

for j in range(1):

   d1=random.randrange(0,m,1)

print("Index of the random number is ",lst[d1])
```

OUTPUT:

Let the maximum Limit for generating random numbers be 10

Index of the random number is 7

RESULT:

Thus, the given program is executed successfully.

QUESTION 6:

Map

A secret message needs to be sent. Use the map function to encrypt the message using Caesar cipher.

CODE:

```
def encrypt(text,s):
    result=""
    for i in range(len(text)):
        char=text[i]

if(char.isupper()):
            result+=chr((ord(char)+s-65)%26+65)
        else:
            result+=chr((ord(char)+s-97)%26+97)
    return result
text="ATTACKATONCE"
s=4
print('Text: '+text)
print('shift: '+str(s))
print('cipher: '+encrypt(text,s))
```

OUTPUT:

```
Text: ATTACKATONCE
shift: 4
cipher: EXXEGOEXSRGI
```

RESULT:

Thus, the given program is executed successfully.

QUESTION 7:

Reduce

Given runs scored by 2 players in a series of matches, write a Python program using reduce function to find who is the better player of the two in terms of maintaining consistency. (You need to find SD).

CODE:

```
import functools
import operator
```

```
p1=[45,81,39]
p2=[24,89,12]
av1=functools.reduce(operator.add,p1)
av2=functools.reduce(operator.add,p2)
av1=av1/2
av2=av2/2
print("Player1" if av1>av2 else "player2")
print("\n")
```

OUTPUT:

```
Player1
```

RESULT:

Thus, the given program is executed successfully.

QUESTION 8:

Map+reduce+filter

Given two trending topics and a bunch of tweets, write a Python program to count the number of tweets that contain each topic. You need to do this by putting together map(), reduce() and filter() functions.

CODE:

```
import functools
q=functools.reduce(lambda x,y: x+y, map(lambda x: x+x, filter (lambda x: (x<=4),[1,2,3,4 ,7])))
print(q)
```

OUTPUT:

```
20
```

RESULT:

Thus, the given program is executed successfully.

EX-12: AUTOMATA PROGRAMMING

DFA:

QUESTION 1:

Write a automata code for the Language that accepts all and only those strings that

contain 001

CODE:

```
from automata.fa.dfa import DFA
dfa = DFA(
    states={'q0', 'q1', 'q2', 'q3'},
input_symbols={'0', '1'},
    transitions={
'q0': {'0': 'q1', '1': 'q0'},
'q1': {'0': 'q2', '1': 'q0'},
'q2': {'0': 'q2', '1': 'q3'},
'q3': {'0': 'q3', '1': 'q3'}
},
initial_state='q0',
final_states={'q3'}
)
for i in range(1,4):
num = input("Enter the string :")
    if(dfa.accepts_input(num)):
        print("Accepted")
    else:
        print("Rejected")
```

OUTPUT:

```
Enter the string :1001

Accepted

Enter the string :010101

Rejected
```

RESULT:

Thus, the given program is executed successfully.

QUESTION 2:

Write a automata code for L(M) ={ w | w has an even number of 1s}

CODE:

```
from automata.fa.dfa import DFA
dfa = DFA(
    states={'q0', 'q1', 'q2'},
input_symbols={'0', '1'},
    transitions={
'q0': {'0': 'q0', '1': 'q1'},
'q1': {'0': 'q1', '1': 'q2'},
'q2': {'0': 'q2', '1': 'q1'}
},
initial_state='q0',
final_states={'q2'}
)
for i in range(1,4):
num = input("Enter the string :")
    if(dfa.accepts_input(num)):
        print("Accepted")
    else:
        print("Rejected")
```

OUTPUT:

```
Enter the string :10101
Rejected
Enter the string :10010
Accepted
```

RESULT:

Thus, the given program is executed successfully.

QUESTION 3:

Write a automata code for L(M) ={0,1}*

CODE:

```
from automata.fa.dfa import DFA
dfa = DFA(
    states={'q0'},
input_symbols={'0', '1'},
    transitions={
'q0': {'0': 'q0', '1': 'q0'}

},
initial_state='q0',
final_states={'q0'}
)
for i in range(1,8):
num = input("Enter the string :")
    if(dfa.accepts_input(num)):
        print("Accepted")

else:
        print("Rejected")
```

OUTPUT:

```
Enter the string :101

Accepted

Enter the string :001

Accepted

Enter the string :720

Rejected
```

RESULT:

Thus, the given program is executed successfully.

QUESTION 4:

Write a automata code for L(M)=a + aa*b.

CODE:

```
from automata.fa.dfa import DFA
dfa = DFA(
    states={'q0', 'q1', 'q2','q3','q4','q5'},
input_symbols={'a', 'b'},
    transitions={
'q0': {'a': 'q1', 'b': 'q5'},
'q1': {'a': 'q2', 'b': 'q5'},
'q2': {'a': 'q3', 'b': 'q4'},
'q3': {'a': 'q2', 'b': 'q5'},
'q4': {'a': 'q5', 'b': 'q5'},
'q5': {'a': 'q5', 'b': 'q5'}
},
initial_state='q0',
final_states={'q1','q4'}
)
for i in range(1,6):
num = input("Enter the string :")
    if(dfa.accepts_input(num)):
        print("Accepted")
    else:
        print("Rejected")
```

OUTPUT:

```
Enter the string :a
Accepted
Enter the string :aab
Accepted
Enter the string :aaab
```

```
Rejected
Enter the string :abab
Rejected
```

RESULT:

Thus, the given program is executed successfully.

QUESTION 5:

Write a automata code for L(M)={(ab) n | n □ N}

CODE:

```
from automata.fa.dfa import DFA
dfa = DFA(
    states={'q0', 'q1', 'q2','q3'},
input_symbols={'a', 'b'},
    transitions={
'q0': {'a': 'q1', 'b': 'q3'},
'q1': {'a': 'q3', 'b': 'q2'},
'q2': {'a': 'q1', 'b': 'q3'},
'q3': {'a': 'q3', 'b': 'q3'},

},
initial_state='q0',
final_states={'q2'}
)
for i in range(1,6):
num = input("Enter the string :")
    if(dfa.accepts_input(num)):
        print("Accepted")
    else:
        print("Rejected")
```

OUTPUT:

```
Enter the string :ab
Accepted
Enter the string :baba
Rejected
Enter the string :aaabbb
Rejected
Enter the string :abab
Accepted
```

RESULT:

Thus, the given program is executed successfully.

QUESTION 6:

Write a automata code for Let Σ = {0, 1}.

Given DFAs for {}, {ε}, Σ * , and Σ + .

6.1 { } -- (null set):

CODE:

```
from automata.fa.dfa import DFA

# DFA which does not accept any input

dfa = DFA(
                states={'q0', 'q1'},
           in
           input_symbols={'0',
           '1'}, transitions={
                'q0': {'0': 'q0', '1': 'q0'},
                'q1': {'0': 'q1', '1': 'q1'}
               },
          initial_state='q0'
          ,
          final_states={'q
          1'}
        )
          for i in range(1,8):
         num = input("Enter the string
         :")
         if(nfa.accepts_input(num)):
             print("Accepted
        ") else:
            print("Rejected")
```

OUTPUT:

```
Enter the string :101

Rejected

Enter the string :100
```

6.3 Rejected
. { ε} -- Accepts only null character:

CODE:

```
from automata.fa.dfa import DFA

# DFA which does not accept any input

dfa = DFA(
        states={'q0', 'q1'},
        input_symbols={'0',
        '1'}, transitions={
             'q0': {'0': 'q1', '1': 'q1'},
              'q1': {'0': 'q1', '1': 'q1'}
        },
        initial_state='q0'
        ,
        final_states={'q
        0'}
  )

        for i in range(1,6):
         num = input("Enter the string :")
        if(dfa.accepts_input(num)):
              print("Accepted")
         else:
                print("Rejected")
```

OUTPUT:

```
Enter the string :10

Rejected
```

```
Enter the string :111
```

```
Rejected
Enter the string :
Accepted
```

RESULT:

Thus, the given program is executed successfully.

1. Σ * -- accepts all combinations of ‘1’ and ‘0’ including nullstring:

CODE:

```
from automata.fa.dfa import DFA
# DFA which does not accept any input

dfa = DFA(
            states={'q0'},
             input_symbols={'0',
             '1'}, transitions={
                    'q0': {'0': 'q0', '1': 'q0'}
             },
              initial_state='q0
            ',
            final_states={'q0
            '}
   )
   for i in range(1,8):
     num = input("Enter the string
     :")
     if(dfa.accepts_input(num)):
       print("Accepted
     ") else:
       print("Rejected")
```

OUTPUT:

```
Enter the string :101

Accepted

Enter the string :1111

Accepted

Enter the string :34

Rejected
```

RESULT:

Thus, the given program is executed successfull

6.4. Σ + -- accepts all combinations of ‘1’ and ‘0’ excluding null string:

CODE:

```
from automata.fa.dfa import DFA
# DFA which does not accept any input
dfa = DFA(
      states={'q0', 'q1'},
      input_symbols={'0',
      '1'}, transitions={
         'q0': {'0': 'q1', '1': 'q1'},
         'q1': {'0': 'q1', '1': 'q1'}
      },
      initial_state='q0'
      ,
      final_states={'q
      1'}
   )
   for i in range(1,8):
      num = input("Enter the string
      :")
      if(dfa.accepts_input(num)):
         print("Accepted
      ") else:
         print("Rejected")
```

OUTPUT:

```
Enter the string :101
Accepted
Enter the string :111
Accepted
Enter the string :
Rejected
```

NFA:

QUESTION 1:

Write a automata code for the Language that accepts all end with 01

CODE:

```
from automata.fa.nfa import NFA

# NFA which accepts strings that ends with '01'

nfa = NFA(

states={'q0', 'q1', 'q2'},

input_symbols={'0', '1'},

transitions={
'q0': {'0': {'q1', 'q0'}, '1': {'q0'}},
'q1': {'1': {'q2'}},
'q2': {}
},
initial_state='q0',
final_states={'q2'}
)

for i in range(1,4):

num = input("Enter the string :")

    if (nfa.accepts_input(num)):
         print("Accepted")

    else:

        print("Rejected")
```

OUTPUT:

```
Enter the string :1001

Accepted

Enter the string :1010

Rejected
```

RESULT:

Thus, the given program is executed successfully.

QUESTION 2:

Write a automata code for L(M)= a + aa*b + a*b.

CODE:

```
from automata.fa.nfa import NFA

# NFA which accepts strings that ends with '01'

nfa = NFA(

states={'q0', 'q1', 'q2','q3','q4'},

input_symbols={'a', 'b'},

transitions={
'q0': {'a': {'q1', 'q2'}},
'q1': {'a': {'q2','q4'},'b':{'q4'}},
'q2': {'a': {'q2'},'b':{'q3'}},
'q3': {},
'q4': {}
},
initial_state='q0',
final_states={'q1','q3'}
)

for i in range(1,6):

num = input("Enter the string :")

    if (nfa.accepts_input(num)):
         print("Accepted")

    else:

        print("Rejected")
```

OTPUT:

```
Enter the string :a
Accepted
Enter the string :aab
Accepted
Enter the string :baba
Rejected
```

RESULT:

Thus, the given program is executed successfully.