

Ex.No:1

Date: **STRUCTURED PROGRAMMING**

AIM:

To implement the structured programming paradigm using python.

1a .Write a program to develop a payroll calculator for school using python. The user input starts by asking for your name or "0" to quit the program, hours worked and pay rate. Extra salary will be provided for over time according to the decision of the admin.

ALGORITHM:

- Start the program
- Get the input from the user
- Find the regular pay if there is no overtime hours
- Find the overtime hour.
- Find the regular time pay, over time pay and net pay
- Print the results

PROGRAM:

```
user = str
end = "0"
hours = round(40,2)
print("One Stop Shop Payroll Calculator")
while user != end:
    print()
    user = input("Please enter your name or type '0' to quit: ")
    if user == end:
        print("End of Report")
    else:
        hours = (float(input("Please enter hours worked: ", )))
```

```
    payrate =(float(input("Please enter your payrate: $", )))
if hours < 40:
    print("Employee's name: ", user)
    print("Overtime hours: 0")
    print("Overtime Pay: $0.00")
    regularpay = round(hours * payrate, 2)
    print("Gross Pay: $", regularpay)
elif hours > 40:
    overtimehours = round(hours - 40.00,2)
    print("Overtime hours: ", overtimehours)
    print("Employee's name: ", user)
    regularpay = round(hours * payrate,2)
    overtimerate = round(payrate * 1.5, 2)
    overtimepay = round(overtimehours * overtimerate)
    grosspay = round(regularpay+overtimepay,2)
    print("Regular Pay: $", regularpay)
    print("Overtime Pay: $",overtimepay)
    print("Gross Pay: $", grosspay)
```

OUTPUT:

One Stop Shop Payroll Calculator

Please enter your name or type '0' to quit: Brandon

Please enter hours worked: 50

Please enter your payrate: \$10

Overtime hours: 10.0

Employee's name: Brandon

Regular Pay: \$ 500.0

Overtime Pay: \$ 150

Gross Pay: \$ 650.0

Please enter your name or type '0' to quit: Brandon

Please enter hours worked: 30

Please enter your payrate: \$10

Employee's name: Brandon

Overtime hours: 0

Overtime Pay: \$0.00

Gross Pay: \$ 300.0

Please enter your name or type '0' to quit: 0

End of Report

Employee's name: 0

Overtime hours: 0

Overtime Pay: \$0.00

Gross Pay: \$ 300.0

Process finished with exit code 0

1b. Write a program to implement Compound Interest

ALGORITHM :

- Start the program
- Get the input principle, rate, time from the user
- Calculate the compound interest
- Print the compound interest
- Stop

PROGRAM :

```
# Program to Calculate compound interest
```

```
principle=1000
```

```
rate=10.25
```

```
time=5
Amount = principle * (pow((1 + rate / 100), time))
CI = Amount - principle
print("Compound interest is", CI)
```

OUTPUT :

Compound interest is 628.8946267774415

1c. Write a program to reverse a given integer

ALGORITHM :

- Start the program
- Assign the input value to the num variable
- Find the reverse number for the given number using while loop
- Print the reversed number
- Stop

PROGRAM :

```
num = 76542
reverse_number = 0
print("Given Number ", num)
while num > 0:
    reminder = num % 10
    reverse_number = (reverse_number * 10) + reminder
    num = num // 10
print("Revered Number ", reverse_number)
```

OUTPUT :

The reversed Number 24567

1d. Write a program to display the cube of the number up to a given integer

ALGORITHM :

- Start the program
- Assign the input value to the input_number variable
- Find the the cube of a given number number using for loop
- Print the cube
- Stop

PROGRAM :

```
input_number = 6
for i in range(1, input_number + 1):
    print("Current Number is :", i, " and the cube is", (i * i * i))
```

OUTPUT :

```
Current Number is : 1  and the cube is 1
Current Number is : 2  and the cube is 8
Current Number is : 3  and the cube is 27
Current Number is : 4  and the cube is 64
Current Number is : 5  and the cube is 125
Current Number is : 6  and the cube is 216
```

1e. Write a program to find the sum of the series $2 + 22 + 222 + 2222 + \dots$ n terms

ALGORITHM :

- Start the program
- Initialize the values for number of terms and start
- Find the sum of series using for loop
- Print the sum of series

- Stop

PROGRAM :

```
number_of_terms = 5
start = 2
sum = 0
for i in range(0, number_of_terms):
    print(start, end=" ")
    sum += start
    start = (start * 10) + 2
print("\nSum of above series is:", sum)
```

OUTPUT :

Sum of the above series: 24690

1f. Write a program to add two matrices using nested loop

ALGORITHM:

- Start the program
- Initialize the X, Y and result matrices
- Iterate through rows and columns
- Print the result using for loop
- Stop

PROGRAM:

```
X = [[12,7,3],
      [4,5,6],
      [7,8,9]]
Y = [[5,8,1],
      [6,7,3],
```

```

[4,5,9]]
result = [[0,0,0],
          [0,0,0],
          [0,0,0]]
# iterate through rows
for i in range(len(X)):
    # iterate through columns
    for j in range(len(X[0])):
        result[i][j] = X[i][j] + Y[i][j]
for r in result:
    print(r)

```

OUTPUT :

```

[17, 15, 4]
[10, 12, 9]
[11, 13, 18]

```

1g. Write a program to find the following pattern

```

*
* *
* * *
* * * *
* * * * *
* * * *
* * *
* *
*

```

ALGORITHM :

- Start the program

- Get the range using for loop
- Print the pattern
- Stop

PROGRAM :

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

OUTPUT :

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

1h. Write a program to convert kilometer to miles

ALGORITHM :

- Start the program
- Get the input from the user

- Initialize the conversion factor
- Convert to miles
- Stop

PROGRAM :

```
# Taking kilometers input from the user
kilometers = float(input("Enter value in kilometers: "))
conv_fac = 0.621371
# calculate miles
miles = kilometers * conv_fac
print('%0.2f kilometers is equal to %0.2f miles' %(kilometers,miles))
```

OUTPUT :

```
Enter value in kilometers: 3.5
3.50 kilometers is equal
```

RESULT:

Thus the Python program to implement payroll calculator, compound interest, to reverse a given integer, to display a cube of a given number, to find the sum of series, to add two matrices, to find pattern and to convert from kilometer to miles are executed successfully.

Ex.No:2**Date:****PROCEDURAL PROGRAMMING****AIM:**

To write a Python Programs to implement procedural programming paradigm

2a. Write a program to generate list of random numbers using procedure

ALGORITHM :

- Start the program

- Import the random package
- Get the random numbers
- Append the random numbers to the list
- Print the random numbers
- Stop

PROGRAM :

```
from random import randint
def get_random_ints(count, begin, end):
print("get_random_ints start")
list_numbers = []
    for x in range(0, count):
list_numbers.append(randint(begin, end))
print("get_random_ints end")
    return list_numbers
print(type(get_random_ints))
nums = get_random_ints(10, 0, 100)
print(nums)
```

OUTPUT :

```
<class 'function'>
get_random_ints start
get_random_ints end
[4, 84, 27, 95, 76, 82, 73, 97, 19, 90]
```

2b. Write a program to return the largest Fibonacci number smaller than x and the lowest fibonacci number higher than x.

ALGORITHM :

- Start the program

- Get the integer from user
- Find largest Fibonacci Number smaller than x
- Find smallest Fibonacci Number larger than x:
- print largest Fibonacci Number smaller than x
- print smallest Fibonacci Number larger than x
- Stop

PROGRAM :

```
def fib_intervall(x):
    if x < 0:
        return -1
    (old,new) = (0,1)
    while True:
        if new < x:
            (old,new) = (new,old+new)
        else:
            if new == x:
                new = old+new
            return (old, new)
while True:
    x = int(input("Your number: "))
    if x <= 0:
        break
    (lub, sup) = fib_intervall(x)
print("Largest Fibonacci Number smaller than x: " + str(lub))
print("Smallest Fibonacci Number larger than x: " + str(sup))
```

OUTPUT :

Your number: 5

Largest Fibonacci Number smaller than x: 3

Smallest Fibonacci Number larger than x: 8

Your number: 4

Largest Fibonacci Number smaller than x: 3

Smallest Fibonacci Number larger than x: 5

Your number: 9

Largest Fibonacci Number smaller than x: 8

Smallest Fibonacci Number larger than x: 13

Your number: -1

2c.write a Python program that accepts a hyphen-separated sequence of words as input and prints the words in a hyphen-separated sequence after sorting them alphabetically.

ALGORITHM:

- Define hyphensepwords procedure
- Get the input and split it
- Use sort function to sequence it
- Print the result
- Call the function

PROGRAM:

```
def hyphensepwords()  
    items=[n for n in input().split('-')]  
    items.sort()  
    print('-'.join(items))  
hyphensepwords()
```

OUTPUT:

green-red-black-white

black-green-red-white

2d. Write a Python program to make a chain of function decorators (bold, italic, underline etc.).

ALGORITHM:

- Define the make_bold function and wrapped function that is defined inside make_bold to create bold letters .
- Define the make_italic function and wrapped function that is defined inside make_italic to create italic letters .
- Define the make_underline function and wrapped function that is defined inside make_underline to create bold letters .
- Call make_bold, make_italic and make_underline functions

PROGRAM:

```
def make_bold(fn):  
    def wrapped():  
        return "<b>" + fn() + "</b>"  
    return wrapped  
  
def make_italic(fn):  
    def wrapped():  
        return "<i>" + fn() + "</i>"  
    return wrapped  
  
def make_underline(fn):  
    def wrapped():  
        return "<u>" + fn() + "</u>"  
    return wrapped  
  
@make_bold  
@make_italic  
@make_underline  
def hello():
```

```
    return "hello world"
print(hello()) ## returns "<b><i><u>hello world</u></i></b>"
```

OUTPUT:

hello world

2e. Write a Python program to access a function inside a function.

ALGORITHM:

- Define a test function
- Define add function inside test function
- Perform addition operation
- Return the result
- Call the test function
- Print the result

PROGRAM:

```
def test(a):
    def add(b):
        nonlocal a
        a += 1
        return a+b
    return add
func= test(4)
print(func(4))
```

OUTPUT:

9

2f. Write a Python function that takes a list and returns a new list with unique elements of the first list.

ALGORITHM:

- Define unique_list function
- Check the uniqueness of each value
- return the unique value
- print the unique list

PROGRAM:

```
def unique_list(l):  
    x = []  
    for a in l:  
        if a not in x:  
            x.append(a)  
    return x  
print(unique_list([1,2,3,3,3,3,4,5]))
```

OUTPUT:

[1, 2, 3, 4, 5]

2g. Write a Python function to check whether a string is a pangram or not.

ALGORITHM:

- Import string and sys packages
- Define a function ispangram
- Check every letter of alphabet is present
- Call the function and check with the quick brown fox jumps over the lazy dog
- Print the result

PROGRAM:


```
import string, sys
def ispangram(str1, alphabet=string.ascii_lowercase):
    alphaset = set(alphabet)
    return alphaset<= set(str1.lower())
print ( ispangram('The quick brown fox jumps over the lazy dog'))
```

OUTPUT:

True

Ex.No:1

Date:

STRUCTURED PROGRAMMING

AIM:

To implement the structured programming paradigm using python.

1a .Write a program to develop a payroll calculator for school using python. The user input starts by asking for your name or "0" to quit the program, hours worked and pay rate. Extra salary will be provided for over time according to the decision of the admin.

ALGORITHM:

- Start the program
- Get the input from the user
- Find the regular pay if there is no overtime hours
- Find the overtime hour.
- Find the regular time pay, over time pay and net pay
- Print the results

PROGRAM:

```
user = str
end = "0"
hours = round(40,2)
print("One Stop Shop Payroll Calculator")
while user != end:
    print()
    user = input("Please enter your name or type '0' to quit: ")
    if user == end:
        print("End of Report")
    else:
        hours = (float(input("Please enter hours worked: ", )))
        payrate =(float(input("Please enter your payrate: $", )))
    if hours < 40:
```

```
print("Employee's name: ", user)
print("Overtime hours: 0")
print("Overtime Pay: $0.00")
regularpay = round(hours * payrate, 2)
print("Gross Pay: $", regularpay)
elif hours > 40:
    overtimehours = round(hours - 40.00,2)
    print("Overtime hours: ", overtimehours)
    print("Employee's name: ", user)
    regularpay = round(hours * payrate,2)
    overtime rate = round(payrate * 1.5, 2)
    overtimepay = round(overtimehours * overtime rate)
    grosspay = round(regularpay+overtimepay,2)
    print("Regular Pay: $", regularpay)
    print("Overtime Pay: $",overtimepay)
    print("Gross Pay: $", grosspay)
```

OUTPUT:

One Stop Shop Payroll Calculator

Please enter your name or type '0' to quit: Brandon

Please enter hours worked: 50

Please enter your payrate: \$10

Overtime hours: 10.0

Employee's name: Brandon

Regular Pay: \$ 500.0

Overtime Pay: \$ 150

Gross Pay: \$ 650.0

Please enter your name or type '0' to quit: Brandon

Please enter hours worked: 30

Please enter your payrate: \$10

Employee's name: Brandon

Overtime hours: 0

Overtime Pay: \$0.00

Gross Pay: \$ 300.0

Please enter your name or type '0' to quit: 0

End of Report

Employee's name: 0

Overtime hours: 0

Overtime Pay: \$0.00

Gross Pay: \$ 300.0

Process finished with exit code 0

1b. Write a program to implement Compound Interest

ALGORITHM :

- Start the program
- Get the input principle, rate, time from the user
- Calculate the compound interest
- Print the compound interest
- Stop

PROGRAM :

```
# Program to Calculate compound interest
```

```
principle=1000
```

```
rate=10.25
```

```
time=5
```

```
Amount = principle * (pow((1 + rate / 100), time))
```

```
CI = Amount - principle  
print("Compound interest is", CI)
```

OUTPUT :

Compound interest is 628.8946267774415

1c. Write a program to reverse a given integer

ALGORITHM :

- Start the program
- Assign the input value to the num variable
- Find the reverse number for the given number using while loop
- Print the reversed number
- Stop

PROGRAM :

```
num = 76542  
reverse_number = 0  
print("Given Number ", num)  
while num > 0:  
    reminder = num % 10  
    reverse_number = (reverse_number * 10) + reminder  
    num = num // 10  
print("Revered Number ", reverse_number)
```

OUTPUT :

The reversed Number 24567

1d. Write a program to display the cube of the number up to a given integer

ALGORITHM :

- Start the program
- Assign the input value to the input_number variable
- Find the the cube of a given number number using for loop
- Print the cube
- Stop

PROGRAM :

```
input_number = 6
for i in range(1, input_number + 1):
    print("Current Number is :", i, " and the cube is", (i * i * i))
```

OUTPUT :

```
Current Number is : 1  and the cube is 1
Current Number is : 2  and the cube is 8
Current Number is : 3  and the cube is 27
Current Number is : 4  and the cube is 64
Current Number is : 5  and the cube is 125
Current Number is : 6  and the cube is 216
```

1e. Write a program to find the sum of the series $2 + 22 + 222 + 2222 + \dots$ n terms

ALGORITHM :

- Start the program
- Initialize the values for number of terms and start
- Find the sum of series using for loop
- Print the sum of series
- Stop

PROGRAM :

```
number_of_terms = 5
start = 2
sum = 0
for i in range(0, number_of_terms):
    print(start, end=" ")
    sum += start
    start = (start * 10) + 2
print("\nSum of above series is:", sum)
```

OUTPUT :

Sum of the above series: 24690

1f. Write a program to add two matrices using nested loop

ALGORITHM:

- Start the program
- Initialize the X,Y and result matrices
- Iterate through rows and columns
- Print the result using for loop
- Stop

PROGRAM:

```
X = [[12,7,3],
      [4 ,5,6],
      [7 ,8,9]]
Y = [[5,8,1],
      [6,7,3],
      [4,5,9]]
result = [[0,0,0],
```

```

        [0,0,0],
        [0,0,0]]
# iterate through rows
for i in range(len(X)):
    # iterate through columns
    for j in range(len(X[0])):
        result[i][j] = X[i][j] + Y[i][j]
for r in result:
    print(r)

```

OUTPUT :

```

[17, 15, 4]
[10, 12, 9]
[11, 13, 18]

```

1g. Write a program to find the following pattern

```

*
* *
* * *
* * * *
* * * * *
* * * *
* * *
* *
*

```

ALGORITHM :

- Start the program
- Get the range using for loop
- Print the pattern

- Stop

PROGRAM :

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

OUTPUT :

```

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

1h. Write a program to convert kilometer to miles

ALGORITHM :

- Start the program
- Get the input from the user
- Initialize the conversion factor
- Convert to miles

- Stop

PROGRAM :

```
# Taking kilometers input from the user
kilometers = float(input("Enter value in kilometers: "))
conv_fac = 0.621371
# calculate miles
miles = kilometers * conv_fac
print('%0.2f kilometers is equal to %0.2f miles' %(kilometers,miles))
```

OUTPUT :

```
Enter value in kilometers: 3.5
3.50 kilometers is equal
```

RESULT:

Thus the Python program to generate list of random numbers, to return the largest Fibonacci number smaller than x and the lowest fibonacci number higher than x, to accept hyphen separated words as input and print the words in hyphen separated words after sorting alphabetically , to create chain of function separators, to access function inside the function, to return unique elements of list from existing list have been executed successfully.

Ex.No:3

Date:

OBJECT ORIENTED PROGRAMMING

AIM:

To Write a Python Programs to implement Object Oriented Programming Paradigm

3a. Write a program to create bank account and to perform deposit and withdraw operations using class and objects

ALGORITHM:

- Start the program
- Create class named Bank Account
- Initialize the constructor to make the balance zero
- Define and implement the withdraw operation.
- Define and implement the deposit operation.
- Create the object
- Call the withdraw and deposit function using object
- Stop

PROGRAM:

```
class BankAccount:
```

```
    def __init__(self):
```

```
        self.balance = 0
```

```
def withdraw(self, amount):  
self.balance -= amount  
return self.balance
```

```
def deposit(self, amount):  
self.balance += amount  
return self.balance
```

```
a = BankAccount()  
b = BankAccount()  
a.deposit(100)  
b.deposit(50)  
b.withdraw(10)  
a.withdraw(10)
```

OUTPUT :

```
100  
50  
40  
90
```

3b. Write a program to create employee class using constructor and destructor and to get ID, name, gender, city and salary

ALGORITHM:

- Start the program
- Initialize all the values using constructor.
- Initialize the destructor
- Get the input from user.
- Display the data

- Create the object for the employee class
- Call functions using class
- Stop

PROGRAM :

class Employee:

```
def __init__(self): #Constructor
    self.__id = 0
    self.__name = ""
    self.__gender = ""
    self.__city = ""
    self.__salary = 0
    print("Object Initialized.")
```

```
def __del__(self): #Destructor
    print("Object Destroyed.")
```

```
def setData(self):
    self.__id=int(input("Enter Id\t:"))
    self.__name = input("Enter Name\t:")
    self.__gender = input("Enter Gender:")
    self.__city = input("Enter City\t:")
    self.__salary = int(input("Enter Salary:"))
```

```
def __str__(self):
    data =
    "["+str(self.__id)+", "+self.__name+", "+self.__gender+", "+self.__city+", "+str(self.__salary)+"]"
    return data
```

```
def showData(self):
    print("Id\t\t:",self.__id)
    print("Name\t:", self.__name)
    print("Gender\t:", self.__gender)
    print("City\t:", self.__city)
    print("Salary\t:", self.__salary)

def main():
    #Employee Object
    emp=Employee()
    emp.setData()
    emp.showData()
    print(emp)

if __name__=="__main__":
    main()
```

OUTPUT :

Object Initialized.

Enter Id :101

Enter Name :Pankaj

Enter Gender:Male

Enter City :Delhi

Enter Salary:70000

Id : 101

Name : Pankaj

Gender : Male

City : Delhi

Salary : 70000

[101,Pankaj,Male,Delhi,70000]

Object Destroyed.

3c.To create the student class that consists of name, id and age attribute and to create the object of the student, to print attribute name of the object, to reset the value of the age, to print the modified value of age , to print true if the student contains the attribute with name and to delete the attribute age.

ALGORITHM :

- Start the program
- Create the student class with name , id and age.
- Create the object of the student class.
- Print attribute name of the object.
- Reset the value of attribute age to 23
- Prints the modified value of age
- Delete the attribute's age.
- Stop

PROGRAM :

class Student:

```
    def __init__(self, name, id, age):
        self.name = name
        self.id = id
    self.age = age
    # creates the object of the class Student
    s = Student("John", 101, 22)
    # prints the attribute name of the objects
    print(getattr(s, 'name'))
    # reset the value of attribute age to 23
    setattr(s, "age", 23)
```

```
# prints the modified value of age
print(getattr(s, 'age'))
print(hasattr(s, 'id'))
# deletes the attribute age
delattr(s, 'age')
```

OUTPUT :

John

23

True

AttributeError: 'Student' object has no attribute 'age'

3d.To implement the object oriented concepts. There are 258 computers available in computer programming lab where each computer is used eight hours per day. Write a Python program using classes and objects that contain `getDetail()` for getting input from user, `calculateSecondsPerDay()` for calculating the usage of each computer in seconds per day, `calculateMinutesPerWeek()` for calculating the usage of each computer in minutes per week, `calculateHoursPerMonth()` for calculating usage of each computer in hour per month and `calculateDaysPerYear()` for calculating usage of each computer in day per year. List all the Components of structured programming language.

ALGORITHM :

- Start the program
- Create the `calc` class with `getdetail()` to get hours.
- Define `calculateSecondsPerDay()` function to calculate seconds per day.
- Define `calculateMinutesPerWeek()` function to calculate minutes in a week.
- Create `calculateDaysPerYear()` function to calculate no. of days in a year.
- Define `calculateHoursPerMonth()` function to compute hours per month.
- Define an object and call the functions

- Stop

PROGRAM :

```
class calc:
    def getDetail(self):
self.total_computer=258
self.total_hour=6
    def calculatesecondsperDay(self):
Second_per_Day=self.total_hour*60*60
print('Total Seconds per Day:',Second_per_Day)
    def calculateminutesperWeek(self):
Minutes_per_Week=self.total_hour*60*7
print("Total Minutes per Week:",Minutes_per_Week)
    def calculatehourperMonth(self):
Hour_per_Month=self.total_hour*30
print("Total Hour per Month:",Hour_per_Month)
    def calculatedayperyear(self):
Day_per_Year=(self.total_hour*365)/24
print("Total Day per Year:",Day_per_Year)
to=calc()
to.getDetail()
to.calculatesecondsperDay()
to.calculateminutesperWeek()
to.calculatehourperMonth()
to.calculatedayperyear()
```

OUTPUT :

Total Seconds per Day: 28800

Total Minutes per Week: 3360

Total Hour per Month: 240

Total Day per Year: 121.66666666666667

RESULT:

Thus the Python program to implement objects oriented concepts have been written and executed successfully.

Ex.No:4

Date:

IMPLEMENTATION OF EVENT DRIVEN PROGRAMMING PARADIGM

AIM:

To implement various kind of events using mouse and keyboard

4a. Program to implement left click and right click events

ALGORITHM :

- Import the package tkinter
- Define the right click event

- Print the right click event detail.
- Define the left click event
- Print the left click event detail.
- Define the middle click event
- Print the middle click event detail.
- Binding the event with buttons
- Create object and run main loop.
- Stop

PROGRAM :

```
From tkinter import *  
root = Tk()  
defrightclick(ev):  
    print("rightclick")  
defleftclick(ev):  
    print("leftclick")  
defmiddleclick(event):  
    print("middleclick")  
frame = Frame(root,width=300,height=200)  
frame.bind("<Button-1>",leftclick)  
frame.bind("<Button-2>",middleclick)  
frame.bind("<Button-3>",rightclick)  
frame.pack()  
root.mainloop()
```

OUTPUT :

```
leftclick  
rightclick  
leftclick
```

rightclick

leftclick

leftclick

4b.Program to implement mouse events

ALGORITHM :

- Import tkinter package
- Create class App
- Initialize the constructor
- Set the frame with green color
- Bind button, double button and button release events
- Bind motion ,enter and leave events
- Create object and run main loop.
- Stop

PROGRAM :

Import tkinter as tk

```
class App(tk.Tk):
```

```
def __init__(self):
```

```
    super().__init__()
```

```
    frame = tk.Frame(self, bg="green", height=100, width=100)
```

```
    frame.bind("<Button-1>", self.print_event)
```

```
    frame.bind("<Double-Button-1>", self.print_event)
```

```
    frame.bind("<ButtonRelease-1>", self.print_event)
```

```
    frame.bind("<B1-Motion>", self.print_event)
```

```
    frame.bind("<Enter>", self.print_event)
```

```
    frame.bind("<Leave>", self.print_event)
```

```
    frame.pack(padx=50, pady=50)
```

```

def print_event(self, event):
    position = "(x={}, y={})".format(event.x, event.y)
    print(event.type, "event", position)

if __name__ == "__main__":
    app = App()
    app.mainloop()

```

OUTPUT :

```

Enter event (x=86, y=1)
Leave event (x=46, y=100)
Enter event (x=48, y=95)
Leave event (x=7, y=100)
Enter event (x=50, y=98)
Leave event (x=47, y=103)
Enter event (x=56, y=95)
Leave event (x=115, y=122)

```

4c. Program to capture keyboard events

ALGORITHM :

- Import tkinter package
- Define Key press and click events
- Bind the keypress and button events
- Create object and run main loop.
- Stop

PROGRAM :

```

From Tkinter import *

root = Tk()

```

```
defkey(event):  
    print"pressed", repr(event.char)  
defcallback(event):  
    frame.focus_set()  
    print"clicked at", event.x, event.y  
    frame = Frame(root, width=100, height=100)  
    frame.bind("<Key>", key)  
    frame.bind("<Button-1>", callback)  
    frame.pack()  
    root.mainloop()
```

OUTPUT :

```
clicked at 61 21  
pressed 'w'  
pressed 'w'  
pressed 'b'  
pressed 'b'  
pressed 'b'  
pressed 'b'  
pressed 'b'  
pressed 'b'  
clicked at 47 54  
clicked at 47 54  
clicked at 47 54
```

4d. Program to implement the keypress event

ALGORITHM :

- Import tkinter package
- Define class App
- Define focus in and key events
- Bind the events
- Create object and run main loop.
- Stop

PROGRAM :

```
Import tkinter as tk
```

```
class App(tk.Tk):
```

```
def __init__(self):
```

```
super().__init__()
```

```
entry = tk.Entry(self)
```

```
entry.bind("<FocusIn>", self.print_type)
```

```
entry.bind("<Key>", self.print_key)
```

```
entry.pack(padx=20, pady=20)
```

```
def print_type(self, event):
```

```
print(event.type)
```

```
def print_key(self, event):
```

```
args = event.keysym, event.keycode, event.char
```

```
print("Symbol: {}, Code: {}, Char: {}".format(*args))
```

```
if __name__ == "__main__":
```

```
app = App()
```

```
app.mainloop()
```

OUTPUT :

FocusIn

Symbol: s, Code: 83, Char: s

Symbol: d, Code: 68, Char: d

Symbol: a, Code: 65, Char: a

Symbol: Caps_Lock, Code: 20, Char:

Symbol: Caps_Lock, Code: 20, Char:

Symbol: v, Code: 86, Char: v

Symbol: c, Code: 67, Char: c

RESULT:

Thus the Python program to implement various mouse click and keyboard events have been executed successful

Ex.No:5

Date:

IMPLEMENTATION OF DECLARATIVE PROGRAMMING PROGRAM

AIM:

To Write a Python Program to create a company table that consists of name, age, address and salary attributes using sqlite3 and to insert five records into the table and to perform select, display, update and delete operations.

ALGORITHM 1:

- Start the program
- Connect to the database using connect function in sqlite3.
- Create the company table
- Insert the records into the table
- Display the data from table
- Display the all columns from the table
- Update the table
- Delete the records from the database
- Stop

PROGRAM 1:

connect to a database

```
conn = sqlite3.connect('test.db')
```

```
print "Opened database successfully";
```

To Create a table

```
import sqlite3
```

```

conn = sqlite3.connect('test.db')
print ("Opened database successfully")

conn.execute("CREATE TABLE COMPANY (ID INT PRIMARY KEY    NOT NULL,
        NAME            TEXT    NOT NULL,
        AGE            INT    NOT NULL,
        ADDRESS        CHAR(50),
        SALARY         REAL);")
print("Table created successfully")
conn.close()

```

To insert records into a table

```

import sqlite3

conn = sqlite3.connect('test.db')
print ("Opened database successfully")

conn.execute("INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) \
VALUES (1, 'Paul', 32, 'California', 20000.00 );");

conn.execute("INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) \
VALUES (2, 'Allen', 25, 'Texas', 15000.00 );");

conn.execute("INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) \
VALUES (3, 'Teddy', 23, 'Norway', 20000.00 );");

conn.execute("INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) \
VALUES (4, 'Mark', 25, 'Rich-Mond ', 65000.00 );");

```

```
conn.commit()
print ("Records created successfully")
conn.close()
```

To display the data from the table

```
import sqlite3
```

```
conn = sqlite3.connect('test.db')
print("Opened database successfully")
```

```
cursor = conn.execute("SELECT id, name, address, salary from COMPANY")
```

```
for row in cursor:
```

```
print "ID = ", row[0]
```

```
print "NAME = ", row[1]
```

```
print "ADDRESS = ", row[2]
```

```
print "SALARY = ", row[3], "\n"
```

```
print("Operation done successfully")
```

```
conn.close()
```

To display all columns from a database

```
import sqlite3
```

```
conn = sqlite3.connect('test.db')
```

```
print("Opened database successfully")
```

```
conn.execute("""CREATE TABLE COMPANY12345
```

```
(ID INT PRIMARY KEY   NOT NULL,
```

```
NAME      TEXT  NOT NULL,
```

```
AGE       INT   NOT NULL,
```

```

        ADDRESS    CHAR(50),
        SALARY     REAL);")
print("Table created successfully")

conn.execute("INSERT INTO COMPANY12345 (ID,NAME,AGE,ADDRESS,SALARY)\
VALUES (1, 'Paul', 32, 'California', 20000.00 )")

conn.execute("INSERT INTO COMPANY12345 (ID,NAME,AGE,ADDRESS,SALARY) \
VALUES (2, 'Allen', 25, 'Texas', 15000.00 )")

conn.execute("INSERT INTO COMPANY12345 (ID,NAME,AGE,ADDRESS,SALARY) \
VALUES (3, 'Teddy', 23, 'Norway', 20000.00 )")

conn.execute("INSERT INTO COMPANY12345 (ID,NAME,AGE,ADDRESS,SALARY) \
VALUES (4, 'Mark', 25, 'Rich-Mond ', 65000.00 )")

conn.commit()
print("Records created successfully")
cursor = conn.execute("SELECT ID,NAME,AGE,ADDRESS,SALARY from
COMPANY12345")
for row in cursor:
    print("ID = ", row[0])
    print("NAME = ", row[1])
    print("AGE=", row[2])
    print("ADDRESS = ", row[3])
    print("SALARY = ", row[4])

print("Operation done successfully");
conn.close()

```

To update the table

```
import sqlite3

conn = sqlite3.connect('test.db')
print("Opened database successfully")

conn.execute("UPDATE COMPANY set SALARY = 25000.00 where ID = 1")
conn.commit()
print "Total number of rows updated :", conn.total_changes

cursor = conn.execute("SELECT id, name, address, salary from COMPANY")
for row in cursor:
    print "ID = ", row[0]
    print "NAME = ", row[1]
    print "ADDRESS = ", row[2]
    print "SALARY = ", row[3], "\n"

print("Operation done successfully")
conn.close()
```

To perform delete operation

```
import sqlite3

conn = sqlite3.connect('test.db')
print( "Opened database successfully")

conn.execute("DELETE from COMPANY where ID = 2;")
conn.commit()
```

```
print "Total number of rows deleted :", conn.total_changes

cursor = conn.execute("SELECT id, name, address, salary from COMPANY")
for row in cursor:
    print "ID = ", row[0]
    print "NAME = ", row[1]
    print "ADDRESS = ", row[2]
    print "SALARY = ", row[3], "\n"

print "Operation done successfully";
conn.close()
```

OUTPUT :

```
print "Opened database successfully";
```

Opened database successfully

Table created successfully

Opened database successfully

Records created successfully

Opened database successfully

ID = 1

NAME = Paul

ADDRESS = California

SALARY = 20000.0

ID = 2

NAME = Allen

ADDRESS = Texas

SALARY = 15000.0

ID = 3

NAME = Teddy

ADDRESS = Norway

SALARY = 20000.0

ID = 4

NAME = Mark

ADDRESS = Rich-Mond

SALARY = 65000.0

Operation done successfully

Opened database successfully

Total number of rows updated : 1

ID = 1

NAME = Paul

ADDRESS = California

SALARY = 25000.0

ID = 2

NAME = Allen

ADDRESS = Texas

SALARY = 15000.0

ID = 3

NAME = Teddy

ADDRESS = Norway

SALARY = 20000.0

ID = 4

NAME = Mark

ADDRESS = Rich-Mond

SALARY = 65000.0

Operation done successfully

Opened database successfully

Total number of rows deleted : 1

ID = 1

NAME = Paul

ADDRESS = California

SALARY = 20000.0

ID = 3

NAME = Teddy

ADDRESS = Norway

SALARY = 20000.0

ID = 4

NAME = Mark

ADDRESS = Rich-Mond

SALARY = 65000.0

Operation done successfully

RESULT:

Thus the company table that consists of name, age, address and salary attributes has been created using sqlite3 and insertion, selection, display, updation and deletion operation have been done successfully.

Ex.No:6**Date:****IMPLEMENTATION OF IMPERATIVE PROGRAMMING PROGRAM****AIM:**

To Write a Python Programs to implement imperative programming paradigm

6a. Write a program to find sum of n numbers

ALGORITHM 1:

- Start the program
- Create the list

- Initialize the sum variable to 0
- Find sum for the numbers in listxxa
- Print the sum
- Stop

PROGRAM1 :

```
my_list = [1, 2, 3, 4, 5]
```

```
sum = 0
```

```
for x in my_list:
```

```
    sum += x
```

```
print(sum)
```

OUTPUT 1:

15

6b. Write a program to get a list of characters and concatenate it to form a string

ALGORITHM :

- Start the program
- Initialize the set of characters
- Get the character one by one and concatenate
- Stop

PROGRAM :

```
sample_characters = ['p','y','t','h','o','n']
```

```
sample_string = ' '
```

```
sample_string = ' '
```

```
sample_string = sample_string + sample_characters[0]
```

```
sample_string = sample_string + 'p'
```

```
sample_string = sample_string + sample_characters[1]
```

```
sample_string = sample_string + 'py'
```

```
sample_string = sample_string + sample_characters[2]
sample_string 'pyt'
sample_string = sample_string + sample_characters[3]
sample_string 'pyth'
sample_string = sample_string + sample_characters[4]
sample_string 'pytho'
sample_string = sample_string + sample_characters[5]
sample_string 'python'
```

OUTPUT 1 :

'python'

6c. Write a program to get a characters 'welcome' and to display

ALGORITHM 2 :

- Start the program
- Initialize the characters in 'welcome'
- Get each string one by one and concatenate each one using for loop
- Print the each character of welcome one by one.
- Stop

PROGRAM 2 :

```
sample_characters = ['w','e','l','c','o','m','e']
sample_string = ""
sample_string
for c in sample_characters:
sample_string = sample_string + c
print(sample_string)
```

OUTPUT 2 :

w

we
wel
welc
welco
welcom
welcome

6d. Write a program to iterate between 10 to 20 and to find factor and to display prime number

ALGORITHM 3 :

- Start the program
- Get the numbers between 10 and 20
- Find the factor of a number
- Display the prime number
- Stop

PROGRAM 3:

```
for num in range(10,20):  
    for i in range(2,num):  
        if num%i == 0:  
            j=num/i  
            print(num,i,j)  
        else:  
            print(num, "is a prime number")
```

OUTPUT 3:

```
10 2 5.0  
10 5 2.0  
10 is a prime number  
11 is a prime number  
12 2 6.0  
12 3 4.0  
12 4 3.0  
12 6 2.0  
12 is a prime number  
13 is a prime number  
14 2 7.0  
14 7 2.0  
14 is a prime number
```

15 3 5.0
15 5 3.0
15 is a prime number
16 2 8.0
16 4 4.0
16 8 2.0
16 is a prime number
17 is a prime number
18 2 9.0
18 3 6.0
18 6 3.0
18 9 2.0
18 is a prime number
19 is a prime number

RESULT:

Thus the Python program to implement to find sum of n numbers, to get a list of characters and concatenate it to form a string, to get a characters 'welcome' and to display and to iterate between 10 to 20 to display prime number have been executed successfully.

Ex.No:7

Date:

IMPLEMENTATION OF PARALLEL PROGRAMMING PROGRAM

AIM:

To Write a Python Programs to

1. To implement parallel processing using multi processing module
2. To implement the thread using thread class
3. To implement synchronization using lock(), acquire() and release method

ALGORITHM 1:

- Start the program
- Import multiprocessing, random and string packages
- Define an output queue
- Generate random numbers
- Setup a list of processes that we want to run
- Run processes
- Exit the completed processes
- Get process results from the output queue
- Print the results
- Stop

PROGRAM 1:

```
import multiprocessing as mp
import random
import string
random.seed(123)
output = mp.Queue()
def rand_string(length, output):
    """ Generates a random string of numbers, lower- and uppercase chars. """
    rand_str = ".join(random.choice(
string.ascii_lowercase
    + string.ascii_uppercase
    + string.digits)
    for i in range(length))
```

```

output.put(rand_str)
processes = [mp.Process(target=rand_string, args=(5, output)) for x in range(4)]
for p in processes:
    p.start()
for p in processes:
    p.join()
results = [output.get() for p in processes]
print(results)

```

OUTPUT 1:

```
['BJWNs', 'GOK0H', '7CTRJ', 'THDF3']
```

ALGORITHM 2:

- Start the program
- Define a function for the thread
- Create two threads using thread function
- Stop

PROGRAM 2 :

```

import thread
import time
def print_time( threadName, delay):
    count = 0
    while count < 5:
        time.sleep(delay)
        count += 1
        print "%s: %s" % ( threadName, time.ctime(time.time()) )

try:

```

```
thread.start_new_thread( print_time, ("Thread-1", 2, ) )
thread.start_new_thread( print_time, ("Thread-2", 4, ) )
except:
    print "Error: unable to start thread"

while 1:
    pass
```

OUTPUT 3 :

```
Thread-1: Thu Jan 22 15:42:17 2009
Thread-1: Thu Jan 22 15:42:19 2009
Thread-2: Thu Jan 22 15:42:19 2009
Thread-1: Thu Jan 22 15:42:21 2009
Thread-2: Thu Jan 22 15:42:23 2009
Thread-1: Thu Jan 22 15:42:23 2009
Thread-1: Thu Jan 22 15:42:25 2009
Thread-2: Thu Jan 22 15:42:27 2009
Thread-2: Thu Jan 22 15:42:31 2009
Thread-2: Thu Jan 22 15:42:35 2009
```

ALGORITHM 4:

- Start the program
- Import threading and time packages
- Create thread using thread class
- Get lock to synchronize threads
- Free lock to release next thread
- Create new threads
- Start new Threads
- Add threads to thread list

- Wait for all threads to complete
- Print the existing main thread
 - Stop

PROGRAM 4:

```
import threading
import time

class myThread (threading.Thread):
    def __init__(self, threadID, name, counter):
        threading.Thread.__init__(self)
        self.threadID = threadID
        self.name = name
        self.counter = counter
    def run(self):
        print "Starting " + self.name
        threadLock.acquire()
        print_time(self.name, self.counter, 3)
        threadLock.release()

def print_time(threadName, delay, counter):
    while counter:
        time.sleep(delay)
        print "%s: %s" % (threadName, time.ctime(time.time()))
        counter -= 1

threadLock = threading.Lock()
threads = []
thread1 = myThread(1, "Thread-1", 1)
thread2 = myThread(2, "Thread-2", 2)
thread1.start()
thread2.start()
```

```
threads.append(thread1)
threads.append(thread2)
for t in threads:
    t.join()
print "Exiting Main Thread"
```

OUTPUT 4:

Starting Thread-1

Starting Thread-2

Thread-1: Thu Mar 21 09:11:28 2013

Thread-1: Thu Mar 21 09:11:29 2013

Thread-1: Thu Mar 21 09:11:30 2013

Thread-2: Thu Mar 21 09:11:32 2013

Thread-2: Thu Mar 21 09:11:34 2013

Thread-2: Thu Mar 21 09:11:36 2013

Exiting Main Thread

RESULT:

Thus the Python program to implement parallel processing using multi processing module, the thread using thread class and synchronization using lock(), acquire() and release method have been executed successfully.

Ex.No:9**Date:****IMPLEMENTATION OF FUNCTIONAL PROGRAMMING PROGRAM****AIM:**

To implement functional programming in python.

1. Write the python program to add two lists of input [7,8,1,2,3,4,5] and [3,4,1,29,1,5,1] using map and lambda and then perform multiplication for the resultant value off add with the input list [4,1,69,2,7,3,6]

ALGORITHM 1:

- Define lambda function to add two numbers and assign a variable.
- Define two lists and pass it to lambda function using map function.
- Print the result
- Create another list and pass the output of previous step.
- Print the output.

PROGRAM 1:

```
add=lambda x,y:x+y
l=[7,8,1,2,3,4,5]
a=[3,4,1,29,1,5,1]
d=list(map(add,l,a))
print(d)
e=[4,1,69,2,7,3,6]
```

```
f=list(map(add,d,e))
print(f)
mul=lambda x,y:x*y
y=list(map(mul,d,f))
print(y)
```

OUTPUT 1:

```
[10, 12, 2, 31, 4, 9, 6]
[14, 13, 71, 33, 11, 12, 12]
[140, 156, 142, 1023, 44, 108, 72]
```

2. Write the python program to return the numbers divisible by 2 from the input list = [21, 24, 12, 34, 10, 15, 41] using lambda and filter function.

ALGORITHM 2:

- Use lambda function to check whether the number is divisible by 4.
- Pass the list as parameter.
- Display the result.

PROGRAM 2 :

```
l=[21,24,12,34,10,15,41]
res=list(filter(lambda x:(x%2==0),l))
print(res)
```

OUTPUT 2 :

```
[24, 12, 34, 10]
```

3. Write the python program to get the multiplication and division from the list of input = [10, 24, 34, 42, 19] using lambda and reduce function.

ALGORITHM 3:

- Define lambda to add two numbers a and b.
- Define lambda to multiply two numbers.
- Import reduce function.
- Pass both the lambda functions to reduce and assign to variables d and e.
- Print the results.

PROGRAM 3 :

```
from functools import reduce
f=[10,24,34,24,19]
d=reduce(lambda a,b:a*b,f)
e=reduce(lambda a,b:a/b,f)
print(d)
print(e)
```

OUTPUT 3:

```
3720960
2.6874785001719986e-05
```

4. Write the python program to find sum of list elements and to find maximum element of the list

ALGORITHM 4:

- Initialize the list
- Use reduce to compute sum of list
- Use reduce to compute maximum element from list
- Print the results.

PROGRAM 4:

```
import functools
lis = [ 1 , 3, 5, 6, 2, ]
print ("The sum of the list elements is : ",end="")
print (functools.reduce(lambda a,b : a+b,lis))
print ("The maximum element of the list is : ",end="")
print (functools.reduce(lambda a,b : a if a > b else b,lis))
```

OUTPUT 4:

he sum of the list elements is : 17
The maximum element of the list is : 6

5. Write the python program to find product of list elements and to find concatenated product of the list

ALGORITHM 5:

- Importing operator for operator functions
- Initialize the list
- Use reduce to compute product
- Use reduce to compute maximum element from list
- Print the results.

PROGRAM 5:

```
import functools
lis = [ 1 , 3, 5, 6, 2, ]
Print ("The product of list elements is : ",end="")
print (functools.reduce(operator.mul,lis))
# using reduce to concatenate string
print ("The concatenated product is : ",end="")
print (functools.reduce(operator.add,["welcome","to","SRM"]))
```

OUTPUT 5:

The product of list elements is : 180

The concatenated product is : welcome to SRM

RESULT:

Thus the Python program to implement functional programming paradigm have been written and executed successfully.

Ex.No:10

Date:

IMPLEMENTATION OF LOGIC PROGRAMMING PARADIGM**AIM:**

To implement logic programming paradigm in python.

1. Write the python program to map values to a function using pyDatalog

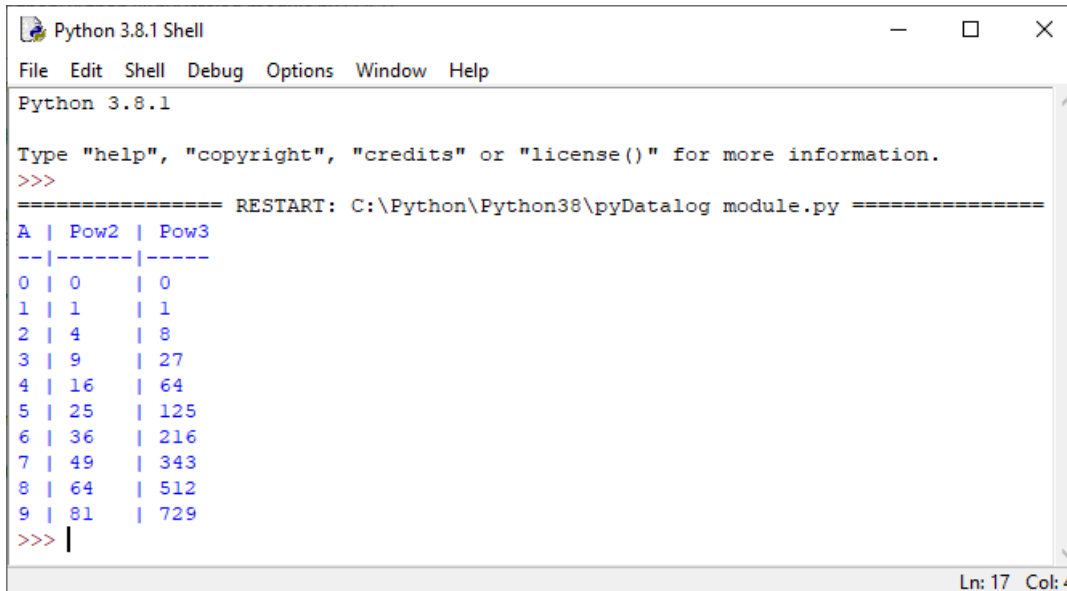
ALGORITHM 1:

- Import pyDatalog and pyDatalog
- Creating new pyDatalog variables
- Creating square function
- Reverse order as desired
- Displaying the output .
- Print the output.

PROGRAM 1:

```
from pyDatalog import pyDatalog
pyDatalog.create_terms("A, Pow2, Pow3")
def square(n):
    return n*n
(predefined logic) as pyDatalogvariable {Optional}
pyDatalog.create_terms("square")
input_values = range(10)[::-1]
(querying with & operator)
print ( A.in_(input_values) & (Pow2 == square(A)) & (Pow3 == A**3) )
```

OUTPUT 1:



```
Python 3.8.1 Shell
File Edit Shell Debug Options Window Help
Python 3.8.1
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Python\Python38\pyDatalog module.py =====
A | Pow2 | Pow3
--|-----|-----
0 | 0     | 0
1 | 1     | 1
2 | 4     | 8
3 | 9     | 27
4 | 16    | 64
5 | 25    | 125
6 | 36    | 216
7 | 49    | 343
8 | 64    | 512
9 | 81    | 729
>>> |
```

Ln: 17 Col: 4

2. Write the python program to solve 8 Queens problem

ALGORITHM 2:

- Import pyDatalog
- The queen in the first column can be in any row
- Find the queens in the first 2 columns, find the first one first, then find a second one
- Repeat for the queens
- The second queen can be in any row, provided it is compatible with the first one
- Find the third queen, first find a queen compatible with the second one, then with the first
- Re-use the previous clause for maximum speed, thanks to memorization
- Repeat for all queens
- Keep one queen in row X1 and another in row X2 if they are separated by N columns
- Print the queens

PROGRAM 2 :

```
from pyDatalog import pyDatalog
```

```
pyDatalog.create_terms('N,X0,X1,X2,X3,X4,X5,X6,X7')
```

```
pyDatalog.create_terms('ok,queens,next_queen')
```

```
queens(X0) <= (X0._in(range(8)))
```

```
queens(X0,X1) <= queens(X0) & next_queen(X0,X1)
```

```
queens(X0,X1,X2) <= queens(X0,X1) & next_queen(X0,X1,X2)
```

```
queens(X0,X1,X2,X3) <= queens(X0,X1,X2) & next_queen(X0,X1,X2,X3)
```

```
queens(X0,X1,X2,X3,X4) <= queens(X0,X1,X2,X3) & next_queen(X0,X1,X2,X3,X4)
```

```
queens(X0,X1,X2,X3,X4,X5) <= queens(X0,X1,X2,X3,X4)
```

```
& next_queen(X0,X1,X2,X3,X4,X5)
```

```
queens(X0,X1,X2,X3,X4,X5,X6) <= queens(X0,X1,X2,X3,X4,X5)
```

```
& next_queen(X0,X1,X2,X3,X4,X5,X6)
```

```
queens(X0,X1,X2,X3,X4,X5,X6,X7) <= queens(X0,X1,X2,X3,X4,X5,X6)
```

```
& next_queen(X0,X1,X2,X3,X4,X5,X6,X7)
```

```
next_queen(X0,X1) <= queens(X1) & ok(X0,1,X1)
```

```
next_queen(X0,X1,X2) <= next_queen(X1,X2) & ok(X0,2,X2)
```

```
next_queen(X0,X1,X2,X3) <= next_queen(X1,X2,X3) & ok(X0,3,X3)
next_queen(X0,X1,X2,X3,X4) <= next_queen(X1,X2,X3,X4) & ok(X0,4,X4)
next_queen(X0,X1,X2,X3,X4,X5) <= next_queen(X1,X2,X3,X4,X5) & ok(X0,5,X5)
next_queen(X0,X1,X2,X3,X4,X5,X6) <= next_queen(X1,X2,X3,X4,X5,X6) & ok(X0,6,X6)
next_queen(X0,X1,X2,X3,X4,X5,X6,X7) <= next_queen(X1,X2,X3,X4,X5,X6,X7) &
ok(X0,7,X7)
ok(X1, N, X2) <= (X1 != X2) & (X1 != X2+N) & (X1 != X2-N)
print(queens(X0,X1,X2,X3,X4,X5,X6,X7).data[0])
```

OUTPUT 2 :

(4, 2, 0, 6, 1, 7, 5, 3)

RESULT:

Thus the Python program to implement logical programming paradigm have been written and executed successfully.

Ex.No:12

Date:

IMPLEMENTATION OF NETWORK PROGRAMMING PARADIGM

AIM:

To implement network programming paradigm in python.

1. Program to implement client server program using TCP

ALGORITHM 1:

TCP Server:

- Import Socket
- Create a socket object
- Reserve a port on your computer
- Next bind to the port this makes the server listen to requests coming from other computers on the network
- Put the socket into listening mode
- Establish connection with client.
- Send a thank you message to the client.
- Close the connection with the client
- Print the output.

TCP Client:

- Import Socket
- Create a socket object
- Define the port on which you want to connect
- Connect to the server on local computer
- Receive data from the server
- Close the connection with the server

- Print the output

PROGRAM 1:

TCP server

```
import socket

s = socket.socket()

print("Socket successfully created")

port = 12345

s.bind(('', port))

print("socket binded to %s" %(port))

s.listen(5)

print("socket is listening")

while True:

    c, addr = s.accept()

    print("Got connection from", addr)

    c.send("Thank you for connecting")

    c.close()
```

OUTPUT :

```
Socket successfully created
socket binded to 12345
socket is listening
Got connection from ('127.0.0.1', 52617)
```

TCP Client

```
# Import socket module
import socket

# Create a socket object
s = socket.socket()

# Define the port on which you want to connect

port = 12345

# connect to the server on local computer

s.connect(('127.0.0.1', port))

# receive data from the server

Print( s.recv(1024) )

# close the connection

s.close()
```

OUTPUT:

Thank you for connecting

Program to implement client server program using UDP

2. Program to implement client server program using UDP

UDP Server:

- Import Socket
- Create a datagram socket
- Bind to address and ip address
- Listen for incoming datagram

- Send reply to client
- Print client message
- Print the client IP address.

UDP Client:

- Import Socket
- Create a client socket
- Create a UDP socket at client side
- Send to server using created UDP socket
- Print the server message

UDP Server:

```
import socket
```

```
localIP = "127.0.0.1"
```

```
localPort = 20001
```

```
bufferSize = 1024
```

```
msgFromServer = "Hello UDP Client"
```

```
bytesToSend = str.encode(msgFromServer)
```

```
UDPServerSocket = socket.socket(family=socket.AF_INET, type=socket.SOCK_DGRAM)
```

```
UDPServerSocket.bind((localIP, localPort))
```

```
Print("UDP server up and listening")
```

```
while(True):
```

```
bytesAddressPair = UDPServerSocket.recvfrom(bufferSize)
```

```
message = bytesAddressPair[0]
```

```
address = bytesAddressPair[1]

clientMsg = "Message from Client: {}".format(message)

clientIP = "Client IP Address: {}".format(address)

print(clientMsg)

print(clientIP)

UDPServerSocket.sendto(bytesToSend, address)
```

OUTPUT:

UDP server up and listening

Message from Client:b"Hello UDP Server"

Client IP Address:("127.0.0.1", 51696)

UDP Client

```
import socket

msgFromClient = "Hello UDP Server"

bytesToSend = str.encode(msgFromClient)

serverAddressPort = ("127.0.0.1", 20001)

bufferSize = 1024

UDPClientSocket = socket.socket(family=socket.AF_INET, type=socket.SOCK_DGRAM)

UDPClientSocket.sendto(bytesToSend, serverAddressPort)

msgFromServer = UDPClientSocket.recvfrom(bufferSize)

msg = "Message from Server {}".format(msgFromServer[0])
```

```
print(msg)
```

OUTPUT:

Message from Server b"Hello UDP Client"

RESULT:

Thus the Python program to implement TCP client and server and UDP client and server have been written and executed successfully.

Ex.No:13**Date:****IMPLEMENTATION OF SYMBOLIC PROGRAMMING PARADIGM****AIM:**

To implement symbolic programming paradigm in python.

13a. Write the commands to perform the operations on substitutions and expressions

ALGORITHM:

- Import sympy module
- Evaluate the expression using sympy command
- Print the result

PROGRAM:

```
from sympy import *  
expr = cos(x) + 1  
print( expr.subs(x, y))  
x, y, z = symbols("x y z")
```

```

print(expr = x**y)
print(expr)
expr = sin(2*x) + cos(2*x)
print( expand_trig(expr))
print(expr.subs(sin(2*x), 2*sin(x)*cos(x)))
expr = x**4 - 4*x**3 + 4*x**2 - 2*x + 3
replacements = [(x**i, y**i) for i in range(5) if i % 2 == 0]
print( expr.subs(replacements))
str_expr = "x**2 + 3*x - 1/2"
expr = sympify(str_expr)
expr
expr.subs(x, 2)
expr = sqrt(8)
print( expr.evalf())
expr = cos(2*x)
expr.evalf(subs={x: 2.4})
one = cos(1)**2 + sin(1)**2
print( (one - 1).evalf())

```

OUTPUT:

```

cos(y)+1
Xxy
2sin(x)cos(x)+2cos2(x)-1
2sin(x)cos(x)+cos(2x)

4x3-2x+y4+4y2+3
192
2.82842712474619

```

0.0874989834394464

$x^2+3x-12$

13b. To perform the following operations on matrices

ALGORITHM:

- Import matrix from sympy.matrices.
- Create the matrix
- Print the matrix
- Display the matrix
- Display 0th row
- Print first column
- Delete the first column from the matrix
- Insert the row into the matrix
- Generate two matrices
- Print addition of two matrices
- Print the multiplication of two matrices

PROGRAM:

```
from sympy.matrices import Matrix
```

```
m=Matrix([[1,2,3],[2,3,1]])
```

```
print( m)
```

```
M=Matrix(2,3,[10,40,30,2,6,9])
```

```
Print( M)
```

```
Print(M.shape)
```

```
Print(M.row(0))
```

```
M.col(1)
```

```
M.row(1)[1:3]
```

```
Print( M)
```

```
M=Matrix(2,3,[10,40,30,2,6,9])
```

```
M.col_del(1)
```

```
a=Matrix([[1,2,3],[2,3,1]])
```

```
print(a)
```

```
a1=Matrix([[10,30]])
```

```
a=M.row_insert(0,M1)
```

```
print(a)
```

```
a2=Matrix([40,6])
```

```
a=M.col_insert(1,M2)
```

```
print(a)
```

```
M1=Matrix([[1,2,3],[3,2,1]])
```

```
M2=Matrix([[4,5,6],[6,5,4]])
```

```
Print( M1+M2)
```

```
M1=Matrix([[1,2,3],[3,2,1]])
```

```
M2=Matrix([[4,5],[6,6],[5,4]])
```

```
Print( M1*M2)
```

OUTPUT:

[1 2 3 2 3 1]

[10 40 30 2 6 9]

(2,3)

[10 40 30]

[40 6]

[6, 9]

[10 30 2 6 9]

Matrix([[10, 30],[2, 9]])

[1 2 3 2 3 1]

[10 40 30 2 9]

[10 40 30 6 9]

[5 7 9 9 7 5]

13c. Write the commands to find derivative, integration, limits, quadratic equation

ALGORITHM:

- Import sympy module
- Make a symbol
- Find the derivative of the expression
- Print the result
- Find the integration of the expression
- Print the result
- Find the limit of the expression

- Print the result
- Find the quadratic equation of the expression
- Print the result

PROGRAM:

```

from sympy import *

x = Symbol('x')

#make the derivative of  $\cos(x)*e^x$ 
ans1 = diff(cos(x)*exp(x), x)

print("The derivative of the  $\sin(x)*e^x$  : ", ans1)

# Compute  $(e^x * \sin(x) + e^x * \cos(x))dx$ 
ans2 = integrate(exp(x)*sin(x) + exp(x)*cos(x), x)

print("The result of integration is : ", ans2)

# Compute definite integral of  $\sin(x^2)dx$ 
# in b / w interval of ? and ?? .

ans3 = integrate(sin(x**2), (x, -oo, oo))

print("The value of integration is : ", ans3)

# Find the limit of  $\sin(x) / x$  given x tends to 0
ans4 = limit(sin(x)/x, x, 0)

print("limit is : ", ans4)

# Solve quadratic equation like, example :  $x^2 - 2 = 0$ 

```

```
ans5 = solve(x**2 - 2, x)
```

```
print("roots are : ", ans5)
```

OUTPUT

```
from sympy import *
```

```
x = Symbol('x')
```

```
#make the derivative of  $\cos(x)*e^x$ 
```

```
ans1 = diff(cos(x)*exp(x), x)
```

```
print("The derivative of the  $\sin(x)*e^x$  : ", ans1)
```

```
# Compute  $(e^x * \sin(x) + e^x * \cos(x))dx$ 
```

```
ans2 = integrate(exp(x)*sin(x) + exp(x)*cos(x), x)
```

```
print("The result of integration is : ", ans2)
```

```
# Compute definite integral of  $\sin(x^2)dx$ 
```

```
# in b / w interval of ? and ?? .
```

```
ans3 = integrate(sin(x**2), (x, -oo, oo))
```

```
print("The value of integration is : ", ans3)
```

```
# Find the limit of  $\sin(x) / x$  given  $x$  tends to 0
```

```
ans4 = limit(sin(x)/x, x, 0)
```

```
print("limit is : ", ans4)
```

```
# Solve quadratic equation like, example :  $x^2 - 2 = 0$ 
```

```
ans5 = solve(x**2 - 2, x)
```

```
print("roots are : ", ans5)
```

RESULT:

Thus the Python program to implement symbolic program have been written and executed successfully.

Ex.No:14**Date:****IMPLEMENTATION OF AUTOMATA PROGRAMMING PARADIGM****AIM:**

To implement symbolic programming paradigm in python.

14a. To write a program to convert DFA to NFA using python.

ALGORITHM:

- Initialize the transitions
- Copy the input in list
- Parse the string of a,b in 0,1 for simplicity
- Counter to remember the number of symbols read
- Set the final states
- Check for each possibility
- Move further only if you are at non-hypothetical state
- Read the last symbol and current state lies in the set of final states
- Input string for next transition is input[i+1:]
- Increment the counter
- Print the state

PROGRAM:

```
import sys

def main():
    transition = [[[0,1],[0]], [[4],[2]], [[4],[3]], [[4],[4]]]
    input = raw_input("enter the string: ")
    input = list(input)
    for index in range(len(input)):
        if input[index]=='a':
            input[index]='0'
        else:
            input[index]='1'

    final = "3"
    i=0
    trans(transition, input, final, start, i)
    print "rejected"
def trans(transition, input, final, state, i):
    for j in range (len(input)):
        for each in transition[state][int(input[j])]:
            if each < 4:
                state = each
                if j == len(input)-1 and (str(state) in final):
                    print "accepted"
```

```

        sys.exit()
    trans(transition, input[i+1:], final, state, i)
    i = i+1
main()

```

OUTPUT:

enter the string: abb
accepted

enter the string: aaaabbbb
rejected

14b. Write a program to convert NFA to DFA

ALGORITHM:

- Take NFA input from User
- Enter total no. of states
- Enter total no. of transitions/paths eg: a,b so input 2 for a,b,c input 3
- Enter state name eg: A, B, C, q1, q2 ..etc
- Creating a nested dictionary
- Enter path eg : a or b in {a,b} 0 or 1 in {0,1}
- Enter all the end states that
- Assign the end states to the paths in dictionary\
- Print NFA
- Enter final state/states of NFA
- Holds all the new states created in dfa
- List of all the paths
- Contains all the states in nfa plus the states created in
- Compute first row of DFA transition table
- Create a nested dictionary in dfa
- Create a single string from all the elements of the list
- Append it to the new_states_list
- Compute the other rows of DFA transition table
- Create a temporary list
- Assign the state in DFA table
- Create a single string(new state) from all the elements of the list
- Assign the new state in the DFA table
- Remove the first element in the new_states_list
- Print the DFA created

- Print Final states of DFA

PROGRAM:

```
import pandas as pd

nfa = {}

n = int(input("No. of states : "))

t = int(input("No. of transitions : "))

for i in range(n):

    state = input("state name : ")

    nfa[state] = {}

    for j in range(t):

        path = input("path : ")

        print("Enter end state from state {} travelling through path {} : ".format(state,path))

        reaching_state = [x for x in input().split()]

        nfa[state][path] = reaching_state

print("\nNFA :- \n")

print(nfa)

print("\nPrinting NFA table :- ")

nfa_table = pd.DataFrame(nfa)

print(nfa_table.transpose())

print("Enter final state of NFA : ")

nfa_final_state = [x for x in input().split()]

new_states_list = []

dfa = {}
```

```

keys_list = list(list(nfa.keys())[0])
path_list = list(nfa[keys_list[0]].keys())
dfa[keys_list[0]] = {}
for y in range(t):
    var = "".join(nfa[keys_list[0]][path_list[y]])
    dfa[keys_list[0]][path_list[y]] = var
    if var not in keys_list:
        new_states_list.append(var)
        keys_list.append(var)
while len(new_states_list) != 0:
    dfa[new_states_list[0]] = {}
    for _ in range(len(new_states_list[0])):
        for i in range(len(path_list)):
            temp = []
            for j in range(len(new_states_list[0])):
                temp += nfa[new_states_list[0][j]][path_list[i]]
            s = ""
            s = s.join(temp)
            if s not in keys_list:
                new_states_list.append(s)
                keys_list.append(s)
            dfa[new_states_list[0]][path_list[i]] = s
    new_states_list.remove(new_states_list[0])
print("\nDFA :- \n")

```

```

print(dfa)

print("\nPrinting DFA table :- ")

dfa_table = pd.DataFrame(dfa)

print(dfa_table.transpose())

dfa_states_list = list(dfa.keys())

dfa_final_states = []

for x in dfa_states_list:

    for i in x:

        if i in nfa_final_state:

            dfa_final_states.append(x)

            break

print("\nFinal states of the DFA are : ",dfa_final_states)

```

OUTPUT:

No. of states : 4

No. of transitions : 2

state name : A

path : a

Enter end state from state A travelling through path a :

A B

path : b

Enter end state from state A travelling through path b :

A

state name : B

path : a

Enter end state from state B travelling through path a :

C

path : b

Enter end state from state B travelling through path b :

C

state name : C

path : a

Enter end state from state C travelling through path a :

D

path : b

Enter end state from state C travelling through path b :

D

state name : D

path : a

Enter end state from state D travelling through path a :

path : b

Enter end state from state D travelling through path b :

NFA :-

{'A': {'a': ['A', 'B'], 'b': ['A']}, 'B': {'a': ['C'], 'b':
['C']}, 'C': {'a': ['D'], 'b': ['D']}, 'D': {'a': [], 'b': []}}

Printing NFA table :-

a b

A [A, B] [A]

B [C] [C]

C [D] [D]

D [] []

Enter final state of NFA :

D

DFA :-

{'A': {'a': 'AB', 'b': 'A'}, 'AB': {'a': 'ABC', 'b': 'AC'},
'ABC': {'a': 'ABCD', 'b': 'ACD'}, 'AC': {'a': 'ABD', 'b':
'AD'}}, 'ABCD': {'a': 'ABCD', 'b': 'ACD'}, 'ACD': {'a': 'ABD',
'b': 'AD'}, 'ABD': {'a': 'ABC', 'b': 'AC'}, 'AD': {'a': 'AB',
'b': 'A'}}

Printing DFA table :-

	a	b
A	AB	A
AB	ABC	AC
ABC	ABCD	ACD
AC	ABD	AD
ABCD	ABCD	ACD
ACD	ABD	AD
ABD	ABC	AC
AD	AB	A

Final states of the DFA are : ['ABCD', 'ACD', 'ABD', 'AD']

RESULT:

Thus the Python program to implement the conversion of DFA to NFA and NFA to DFA have been written and executed successfully.

Ex.No:15**Date:****IMPLEMENTATION OF GUI PROGRAMMING PARADIGM****AIM:**

To implement GUI programming paradigm in python.

15a. Design a calculator to perform all mathematical operations using python

ALGORITHM:

- Import sympy module
- Evaluate the expression using sympy command
- Print the result

PROGRAM:

'btn_click' function continuously updates the input field whenever you enters a number

```
def btn_click(item):
```

```
    global expression
```

```
    expression = expression + str(item)
```

```
    input_text.set(expression)
```

'btn_clear' function clears the input field

```
def btn_clear():
```

```
    global expression
```

```
    expression = " "
```

```
    input_text.set(" ")
```

'btn_equal' calculates the expression present in input field

```
def btn_equal():
```

```
    global expression
```

```
    result = str(eval(expression))
```

'eval' function evaluates the string expression directly

```
    input_text.set(result)
```

```
    expression = " "
```

'StringVar()' is used to get the instance of input field

```
    input_text = StringVar()
```

creating a frame for the input field

```
input_frame = Frame(window, width = 312, height = 50, bd = 0, highlightbackground = "black",  
highlightcolor = "black", highlightthickness = 1)
```

```

input_frame.pack(side = TOP)
# creating a input field inside the 'Frame'
input_field = Entry(input_frame, font = ('arial', 18, 'bold'), textvariable = input_text, width = 50,
bg = "#eee", bd = 0, justify = RIGHT)
input_field.grid(row = 0, column = 0)
input_field.pack(ipady = 10)
# 'ipady' is internal padding to increase the height of input field
# creating another 'Frame' for the button below the 'btns_frame'
btns_frame = Frame(window, width = 312, height = 272.5, bg = "grey")
btns_frame.pack()
# first row
clear = Button(btns_frame, text = "C", fg = "black", width = 32, height = 3, bd = 0, bg = "#eee",
cursor = "hand2", command = lambda: btn_clear()).grid(row = 0, column = 0, columnspan =
3, padx = 1, pady = 1)
divide = Button(btns_frame, text = "/", fg = "black", width = 10, height = 3, bd = 0, bg = "#eee",
cursor = "hand2", command = lambda: btn_click("/")).grid(row = 0, column = 3, padx = 1, pady
= 1)
# second row
seven = Button(btns_frame, text = "7", fg = "black", width = 10, height = 3, bd = 0, bg = "#fff",
cursor = "hand2", command = lambda: btn_click(7)).grid(row = 1, column = 0, padx = 1, pady =
1)
eight = Button(btns_frame, text = "8", fg = "black", width = 10, height = 3, bd = 0, bg = "#fff",
cursor = "hand2", command = lambda: btn_click(8)).grid(row = 1, column = 1, padx = 1, pady =
1)
nine = Button(btns_frame, text = "9", fg = "black", width = 10, height = 3, bd = 0, bg = "#fff",
cursor = "hand2", command = lambda: btn_click(9)).grid(row = 1, column = 2, padx = 1, pady =
1)

```

```
multiply = Button(btns_frame, text = "*", fg = "black", width = 10, height = 3, bd = 0, bg =
"#eee", cursor = "hand2", command = lambda: btn_click("*")).grid(row = 1, column = 3, padx =
1, pady = 1)
# third row
four = Button(btns_frame, text = "4", fg = "black", width = 10, height = 3, bd = 0, bg = "#fff",
cursor = "hand2", command = lambda: btn_click(4)).grid(row = 2, column = 0, padx = 1, pady =
1)
five = Button(btns_frame, text = "5", fg = "black", width = 10, height = 3, bd = 0, bg = "#fff",
cursor = "hand2", command = lambda: btn_click(5)).grid(row = 2, column = 1, padx = 1, pady =
1)
six = Button(btns_frame, text = "6", fg = "black", width = 10, height = 3, bd = 0, bg = "#fff",
cursor = "hand2", command = lambda: btn_click(6)).grid(row = 2, column = 2, padx = 1, pady =
1)
minus = Button(btns_frame, text = "-", fg = "black", width = 10, height = 3, bd = 0, bg = "#eee",
cursor = "hand2", command = lambda: btn_click("-")).grid(row = 2, column = 3, padx = 1, pady
= 1)
# fourth row
one = Button(btns_frame, text = "1", fg = "black", width = 10, height = 3, bd = 0, bg = "#fff",
cursor = "hand2", command = lambda: btn_click(1)).grid(row = 3, column = 0, padx = 1, pady =
1)
two = Button(btns_frame, text = "2", fg = "black", width = 10, height = 3, bd = 0, bg = "#fff",
cursor = "hand2", command = lambda: btn_click(2)).grid(row = 3, column = 1, padx = 1, pady =
1)
three = Button(btns_frame, text = "3", fg = "black", width = 10, height = 3, bd = 0, bg = "#fff",
cursor = "hand2", command = lambda: btn_click(3)).grid(row = 3, column = 2, padx = 1, pady =
1)
plus = Button(btns_frame, text = "+", fg = "black", width = 10, height = 3, bd = 0, bg = "#eee",
cursor = "hand2", command = lambda: btn_click("+")).grid(row = 3, column = 3, padx = 1, pady
= 1)
```

fifth row

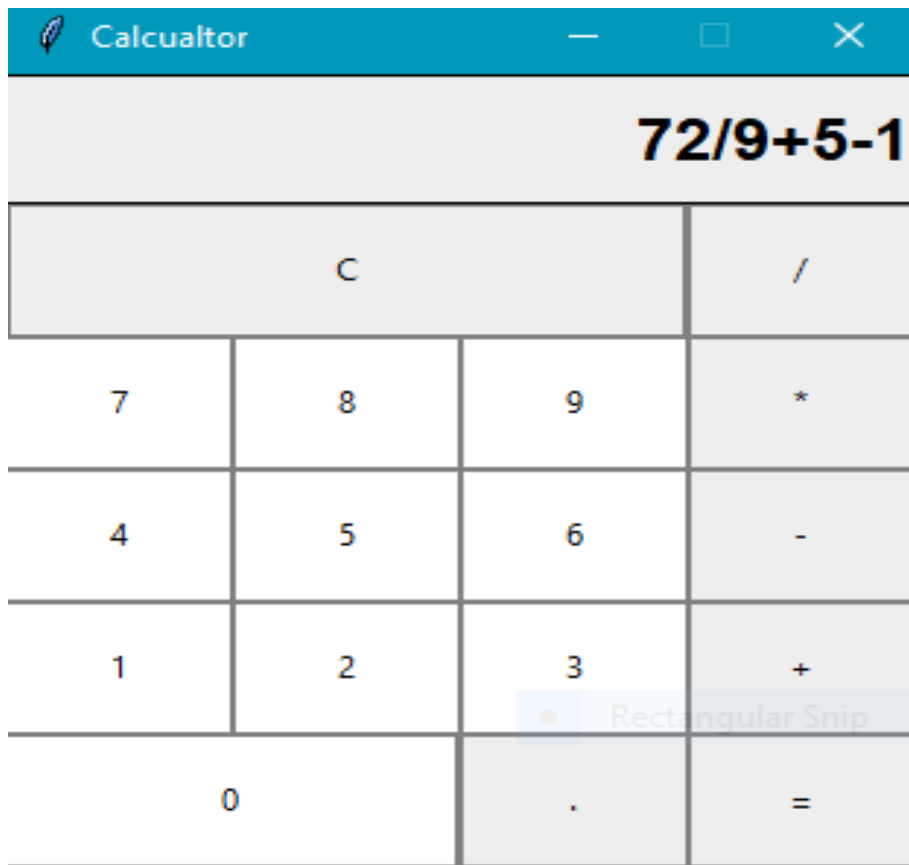
```
zero = Button(btns_frame, text = "0", fg = "black", width = 21, height = 3, bd = 0, bg = "#fff",  
cursor = "hand2", command = lambda: btn_click(0)).grid(row = 4, column = 0, columnspan = 2,  
padx = 1, pady = 1)
```

```
point = Button(btns_frame, text = ".", fg = "black", width = 10, height = 3, bd = 0, bg = "#eee",  
cursor = "hand2", command = lambda: btn_click(".")).grid(row = 4, column = 2, padx = 1, pady  
= 1)
```

```
equals = Button(btns_frame, text = "=", fg = "black", width = 10, height = 3, bd = 0, bg = "#eee",  
cursor = "hand2", command = lambda: btn_equal()).grid(row = 4, column = 3, padx = 1, pady =  
1)
```

```
window.mainloop()
```

OUTPUT



15b. Write a program to implement employee salary calculation using python. Input the employee name and the basic salary calculate and display the net salary using following condition.

(i)if the basic salary is more than 50000 then include 10% tax.

(ii)if the basic salary is more than 30000 then include 5% tax.

ALGORITHM:

- Import matrix from sympy.matrices.
- Create the matrix
- Print the matrix
- Display the matrix
- Display 0th row
- Print first column
- Delete the first column from the matrix
- Insert the row into the matrix
- Generate two matrices
- Print addition of two matrices
- Print the multiplication of two matrices

PROGRAM:

```
from tkinter import *
def Ok():
    result = float(e2.get())
    if(result > 50000) :
        tax = result * 10/100
    elif(result > 30000) :
        tax = result * 5/100
```

```
    else :
        tax = 0
    taxText.set(tax)
    nsal = result - tax
    nsalText.set(nsal)
root = Tk()
root.title("Employee Salary Calculation System")
root.geometry("300x400")
global e1
global e2
global taxText
global nsalText
taxText = StringVar()
nsalText = StringVar()
Label(root, text="Employee Name").place(x=10, y=10)
Label(root, text="Salary").place(x=10, y=40)
Label(root, text="Tax").place(x=10, y=80)
Label(root, text="Total:").place(x=10, y=110)
e1 = Entry(root)
e1.place(x=100, y=10)
e2 = Entry(root)
e2.place(x=100, y=40)
tax = Label(root, text="", textvariable=taxText).place(x=100,y=80)
nsal = Label(root, text="", textvariable=nsalText).place(x=100, y=110)
Button(root, text="Cal", command=Ok ,height = 1, width =3).place(x=10, y=150)
empname = Entry(root)
empsal = Entry(root)
root.mainloop()
```

OUTPUT:

The screenshot shows a window titled "Employee Salary Calculat...". Inside the window, there are two input fields: "Employee Name" with the value "John" and "Salary" with the value "75000". Below these, the calculated values are displayed: "Tax" as "7500.0" and "Total:" as "67500.0". A button labeled "Cal" is located at the bottom left of the input section.

Employee Name	John
Salary	75000
Tax	7500.0
Total:	67500.0

Cal

RESULT:

Thus the Python GUI program for calculator and employee salary calculation have been written and executed successfully.