EXPERIMENT NUMBER –Practical 6.1

STUDENT'S NAME –

STUDENT'S UID -

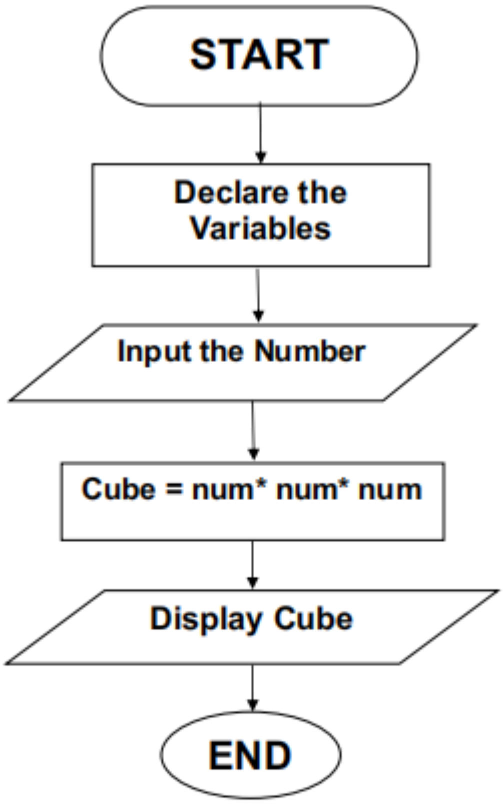
CLASS AND GROUP-

SEMESTER - 2

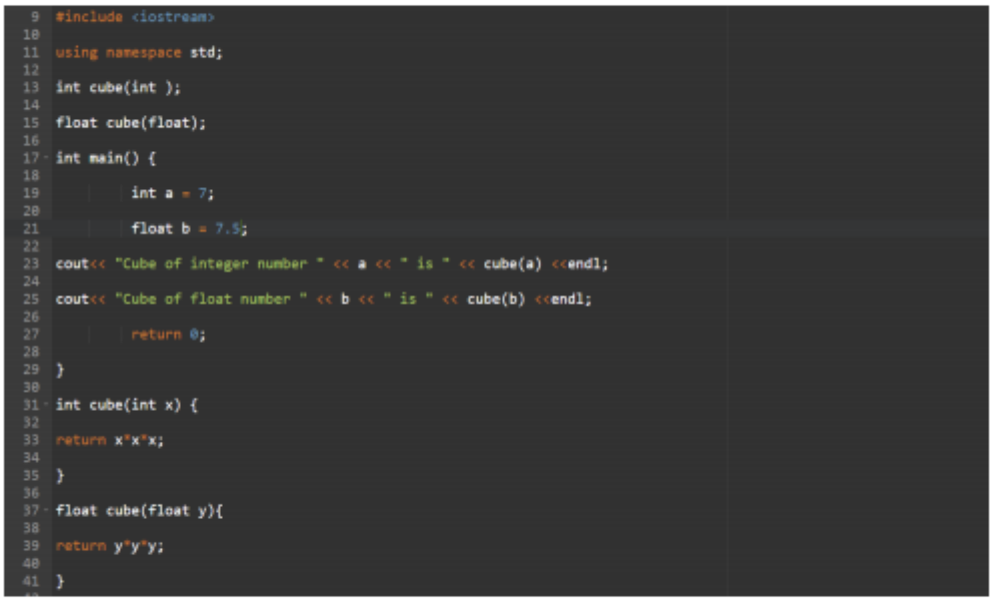
TOPIC OF EXPERIMENT - POLYMORPHISM

AIM OF THE EXPERIMENT - WAP to calculate and display cube of an integer and float variable using function overloading.

FLOWCHART/ ALGORITHM-

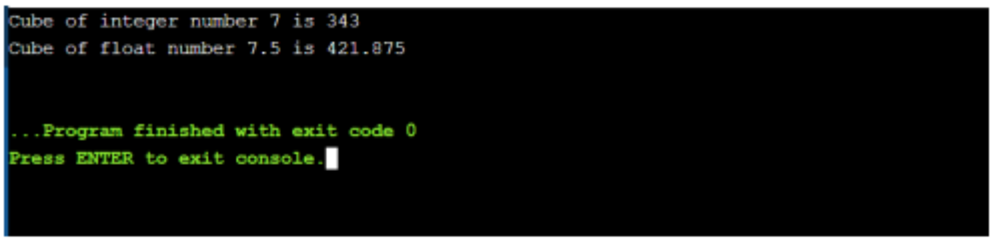


PROGRAM CODE-



```
#include <iostream>

using namespace std;

int cube(int );

float cube(float);

int main() {

        int a = 7;

        float b = 7.5;

cout<< "Cube of integer number " << a << " is " << cube(a) <<endl;

cout<< "Cube of float number " << b << " is " << cube(b) <<endl;

        return 0;

}

int cube(int x) {

return x*x*x;

}

float cube(float y){

return y*y*y;

}
```

OUTPUT -



```
Cube of integer number 7 is 343
Cube of float number 7.5 is 421.875


...Program finished with exit code 0
Press ENTER to exit console.
```

EXPERIMENT NUMBER –Practical 6.2

STUDENT'S NAME - YASH RAJ

STUDENT'S UID - 21BCS11765

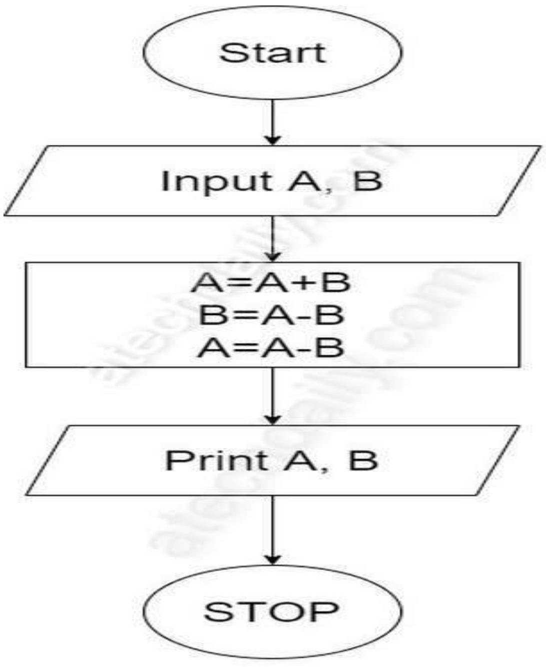
CLASS AND GROUP- 509-B

SEMESTER - 2

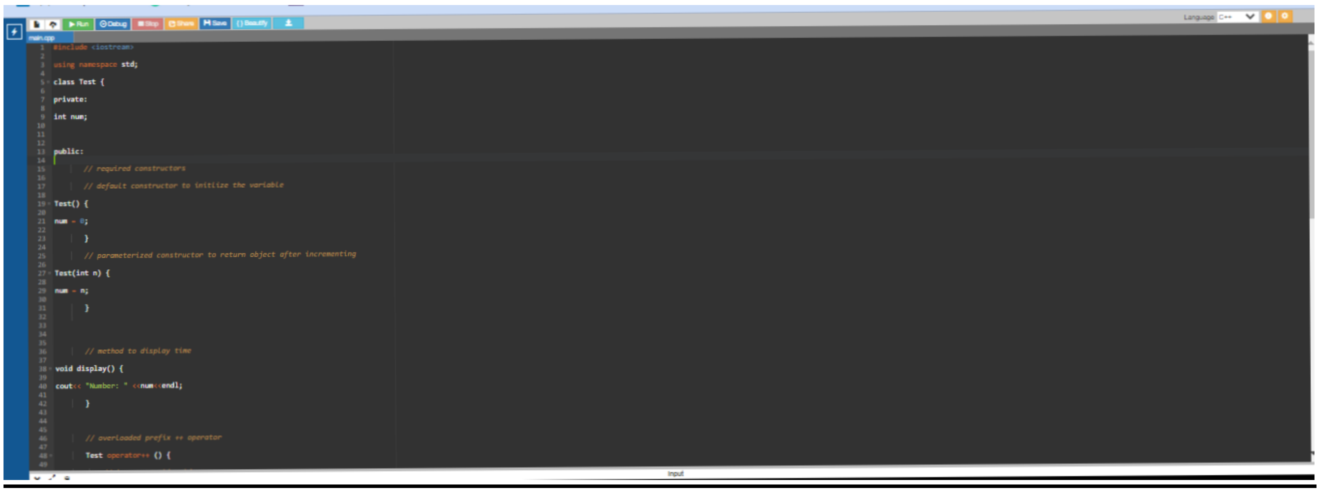
TOPIC OF EXPERIMENT – POLYMORPHISM

AIM OF THE EXPERIMENT - Program to demonstrate the unary operator overloading for operator ++. Make a class test. Create a default constructor to initialize the variable. 1) Overload operator ++ (Pre) with definition to pre-decrement the value of a variable 2) Overload operator ++ (post) with definition to post-decrement the value of variable.

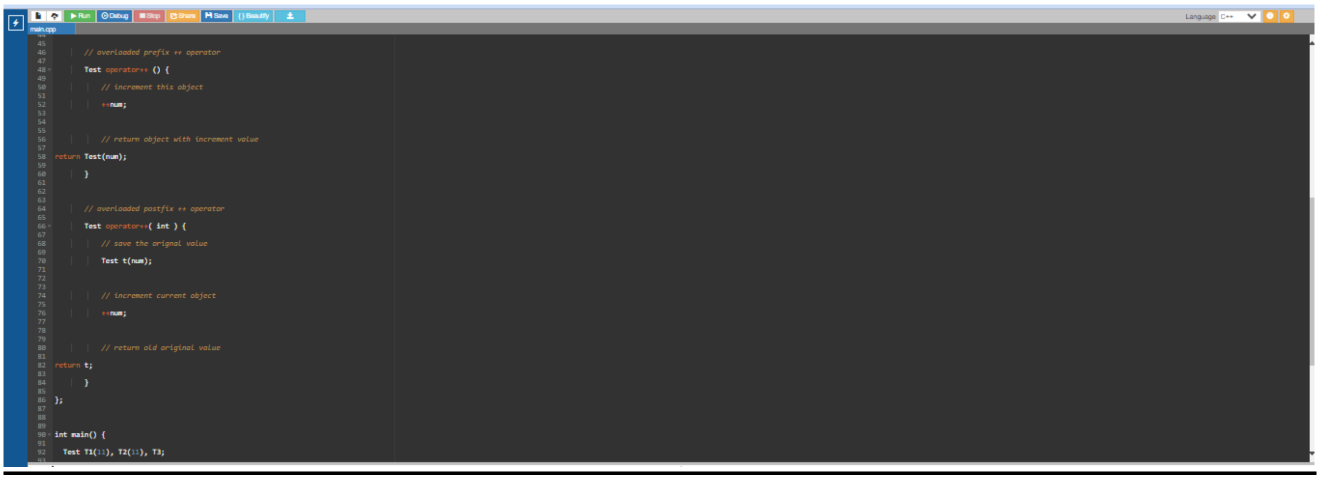
FLOWCHART/ ALGORITHM -



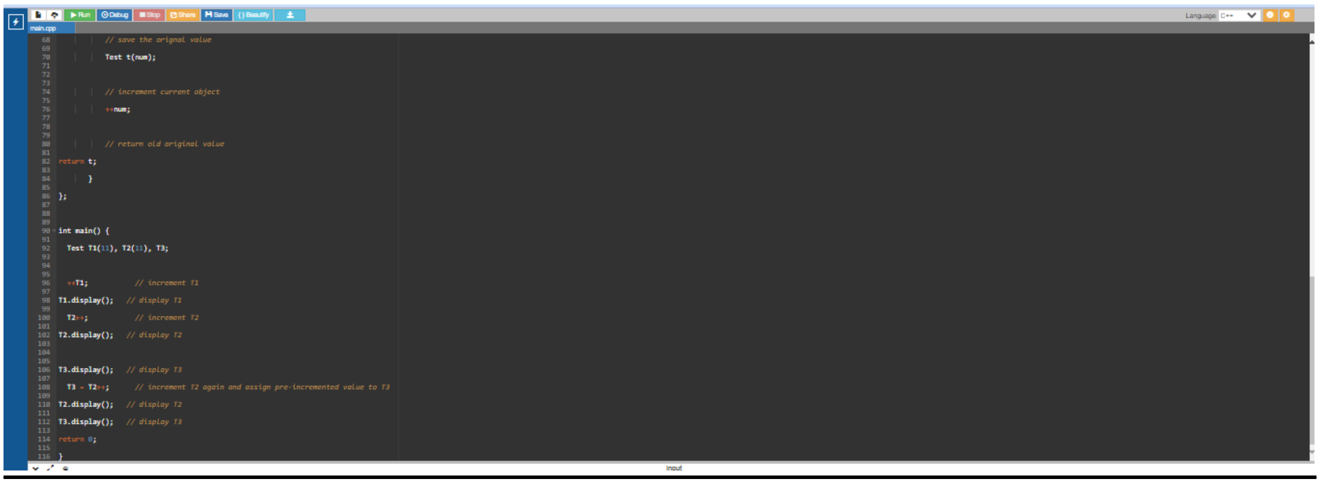
PROGRAM CODE-



```cpp
#include <iostream>

using namespace std;

class Test {

private:

int num;



public:

        // required constructors

        // default constructor to initize the variable

Test() {

num = 0;

        }

        // parameterized constructor to return object after incrementing

Test(int n) {

num = n;

        }




        // method to display time

void display() {

cout<< "Number: " <<num<<endl;

        }



        // overloaded prefix ++ operator

        Test operator++ () {
```

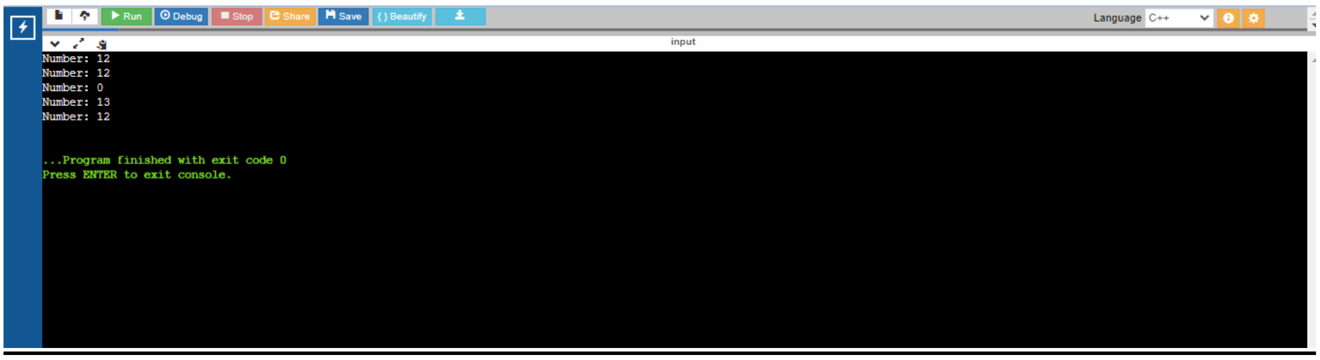


```cpp

        // overloaded prefix ++ operator

        Test operator++ () {

            // increment this object

            ++num;



            // return object with increment value

return Test(num);

        }



        // overloaded postfix ++ operator

        Test operator++( int ) {

            // save the orignal value

            Test t(num);



            // increment current object

            ++num;



            // return old original value

return t;

        }

};



int main() {

  Test T1(11), T2(11), T3;
```



```cpp
            // save the orignal value

            Test t(num);



            // increment current object

            ++num;



            // return old original value

return t;

        }

};



int main() {

  Test T1(11), T2(11), T3;



  ++T1;           // increment T1

T1.display();   // display T1

  T2++;           // increment T2

T2.display();   // display T2



T3.display();   // display T3

  T3 = T2++;      // increment T2 again and assign pre-incremented value to T3

T2.display();   // display T2

T3.display();   // display T3

return 0;

}
```

OUTPUT-



```
Number: 12
Number: 12
Number: 0
Number: 13
Number: 12


...Program finished with exit code 0
Press ENTER to exit console.
```

EXPERIMENT NUMBER –Practical 6.3

STUDENT'S NAME – YASH RAJ

STUDENT'S UID – 21BCS11765

CLASS AND GROUP – 509-B

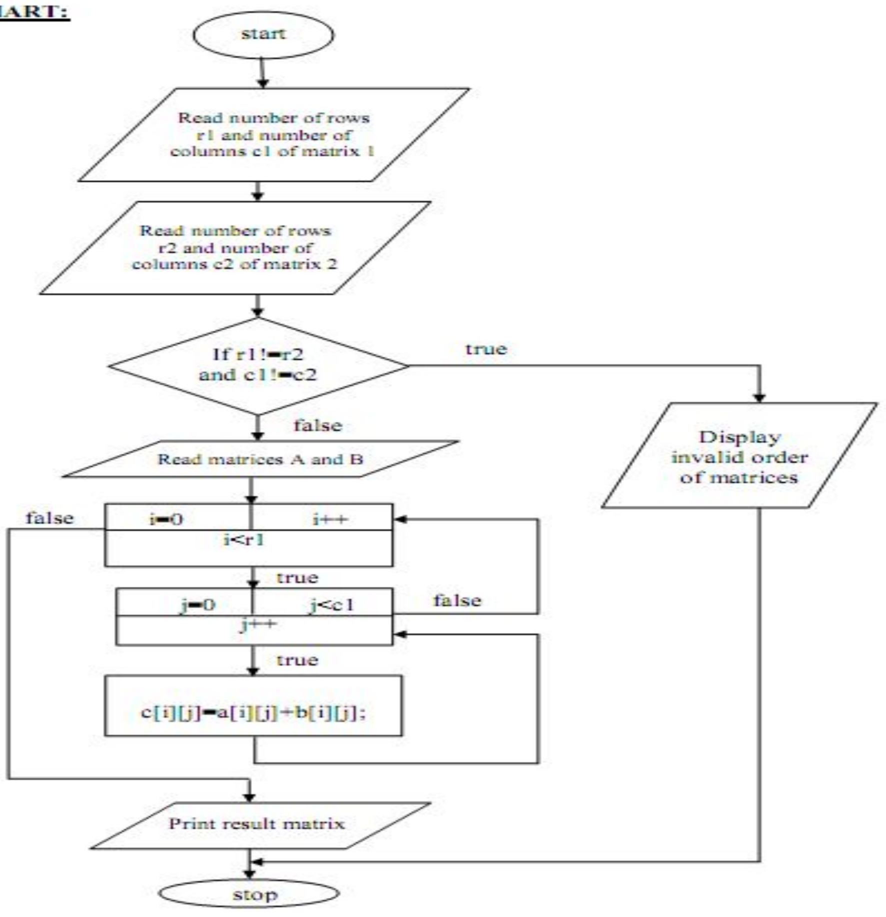
SEMESTER – 2ND

TOPIC OF EXPERIMENT – POLYMORPHISM

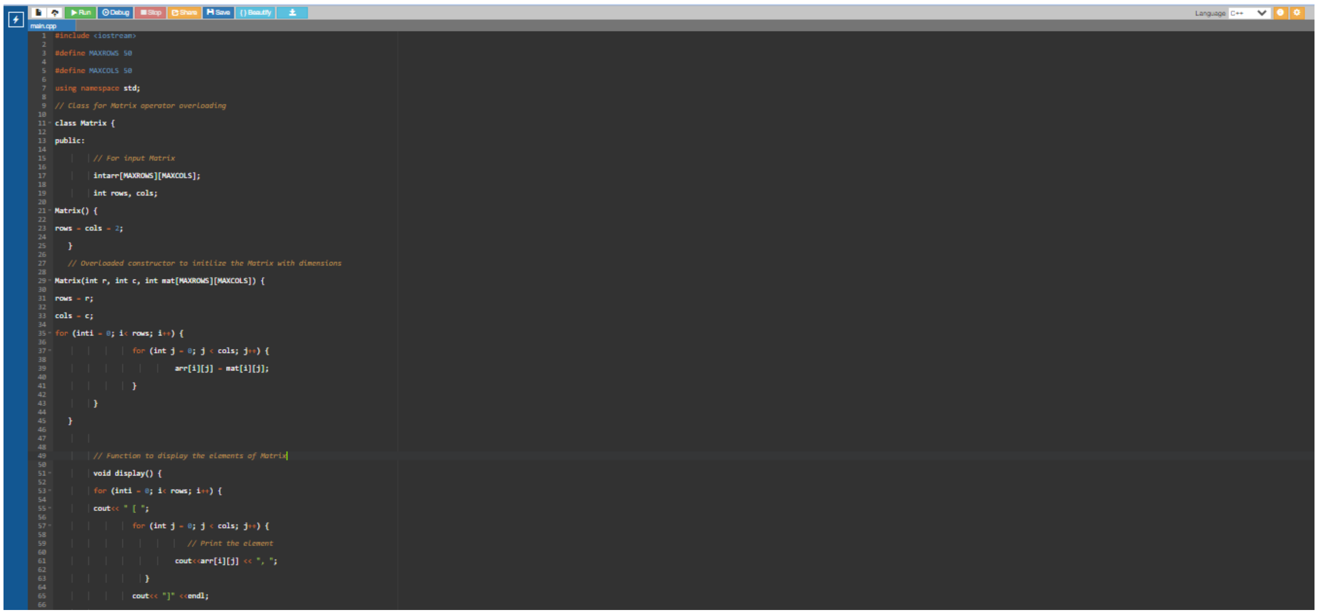
AIM OF THE EXPERIMENT - WAP for creating a matrix class which can handle integer matrices of different dimensions. Overload the operator (+) for addition and (==) comparison of matrices.

FLOWCHART/ ALGORITHM -

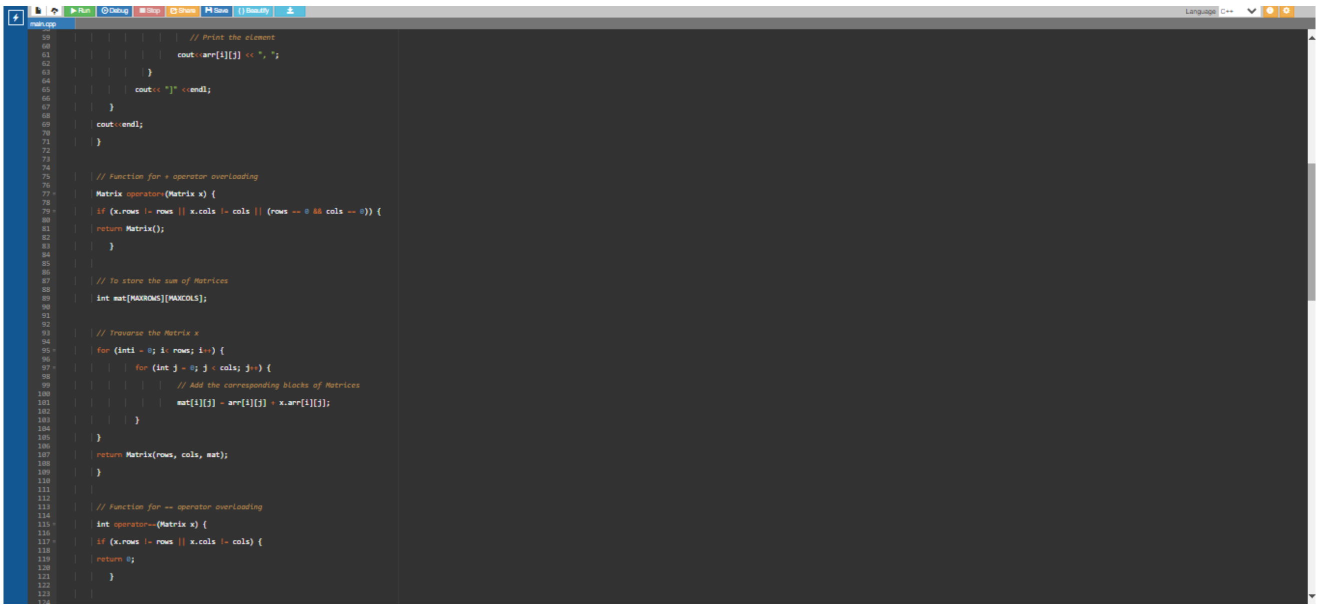
FLOWCHART:


start
Read number of rows r1 and number of columns c1 of matrix 1
Read number of rows r2 and number of columns c2 of matrix 2
If r1 != r2 and c1 != c2
true
Display invalid order of matrices
false
Read matrices A and B
false
i=0
i++
i<r1
true
j=0
j<c1
j++
false
true
c[i][j]=a[i][j]+b[i][j];
Print result matrix
stop

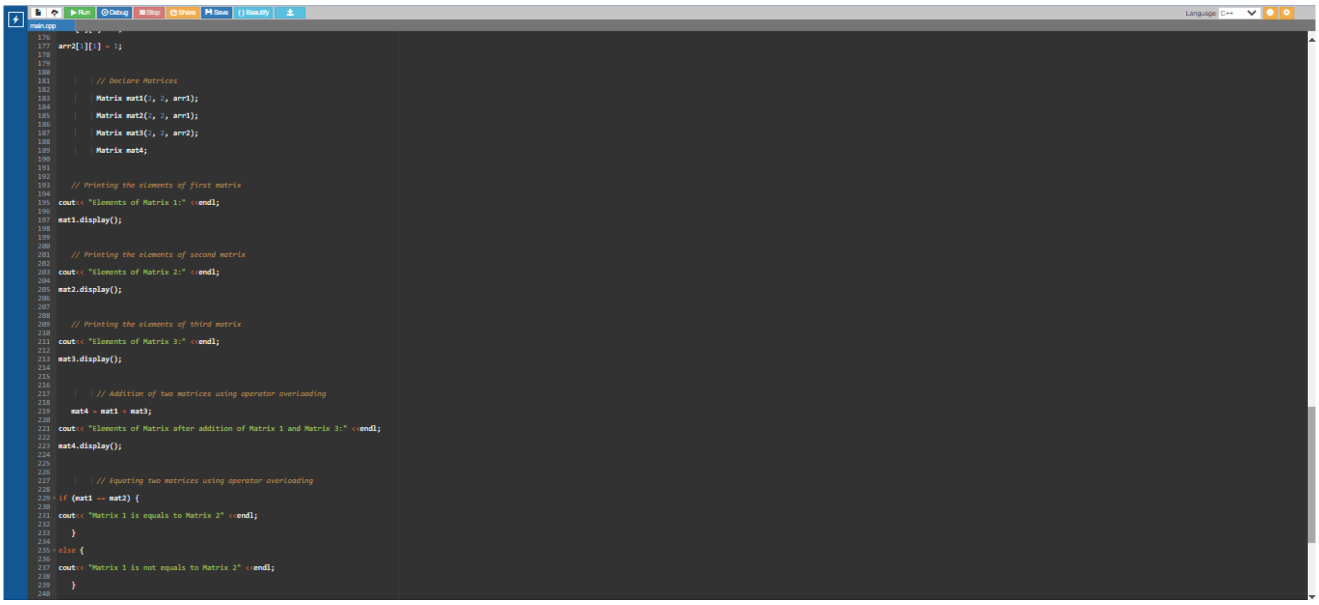
PROGRAM CODE-



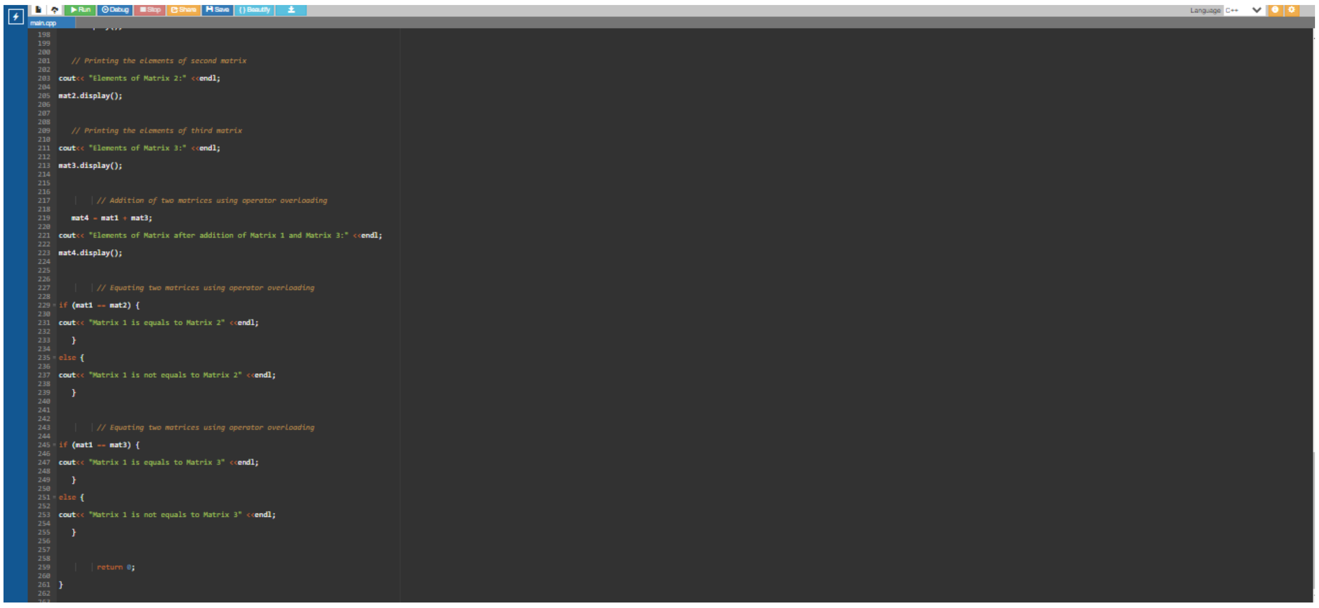
```
#include <iostream>

#define MAXROWS 50

#define MAXCOLS 50

using namespace std;

// Class for Matrix operator overloading

class Matrix {

public:

        // For input Matrix

        intarr[MAXROWS][MAXCOLS];

        int rows, cols;

Matrix() {

rows = cols = 2;

    }

    // Overloaded constructor to initlize the Matrix with dimensions

Matrix(int r, int c, int mat[MAXROWS][MAXCOLS]) {

rows = r;

cols = c;

for (inti = 0; i< rows; i++) {

                  for (int j = 0; j < cols; j++) {

                            arr[i][j] = mat[i][j];

                  }

        }

    }



        // Function to display the elements of Matrix

        void display() {

        for (inti = 0; i< rows; i++) {

        cout<< " [ ";

                  for (int j = 0; j < cols; j++) {

                                // Print the element

                            cout<<arr[i][j] << ", ";

                     }

                  cout<< "]" <<endl;
```



```
                                // Print the element

                            cout<<arr[i][j] << ", ";

                     }

                  cout<< "]" <<endl;

            }

        cout<<endl;

        }



        // Function for + operator overloading

        Matrix operator+(Matrix x) {

        if (x.rows != rows || x.cols != cols || (rows == 0 && cols == 0)) {

        return Matrix();

            }



        // To store the sum of Matrices

        int mat[MAXROWS][MAXCOLS];



        // Traverse the Matrix x

        for (inti = 0; i< rows; i++) {

                  for (int j = 0; j < cols; j++) {

                            // Add the corresponding blocks of Matrices

                            mat[i][j] = arr[i][j] + x.arr[i][j];

                  }

        }

        return Matrix(rows, cols, mat);

        }



        // Function for == operator overloading

        int operator==(Matrix x) {

        if (x.rows != rows || x.cols != cols) {

        return 0;

            }


```

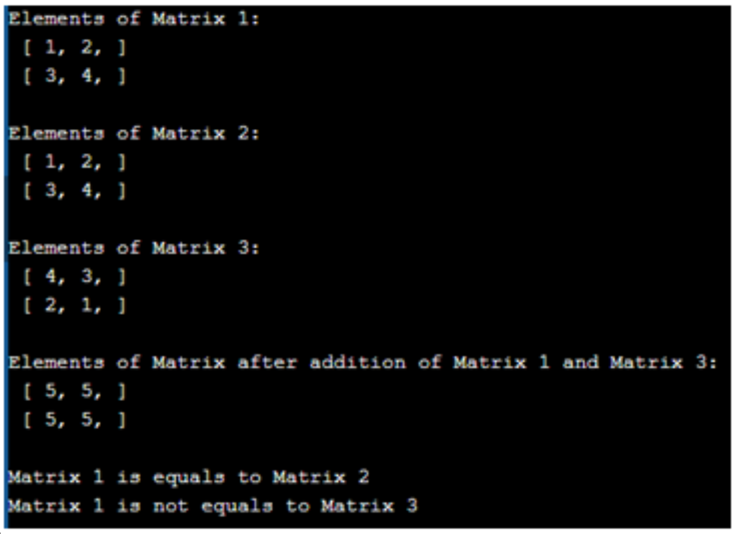


```cpp
arr2[1][1] = 1;


        // Declare Matrices

        Matrix mat1(2, 2, arr1);

        Matrix mat2(2, 2, arr1);

        Matrix mat3(2, 2, arr2);

        Matrix mat4;


    // Printing the elements of first matrix

cout<< "Elements of Matrix 1:" <<endl;

mat1.display();


    // Printing the elements of second matrix

cout<< "Elements of Matrix 2:" <<endl;

mat2.display();


    // Printing the elements of third matrix

cout<< "Elements of Matrix 3:" <<endl;

mat3.display();


        // Addition of two matrices using operator overloading

    mat4 = mat1 + mat3;

cout<< "Elements of Matrix after addition of Matrix 1 and Matrix 3:" <<endl;

mat4.display();


        // Equating two matrices using operator overloading

if (mat1 == mat2) {

cout<< "Matrix 1 is equals to Matrix 2" <<endl;

    }

else {

cout<< "Matrix 1 is not equals to Matrix 2" <<endl;

    }
```



```cpp
    // Printing the elements of second matrix

cout<< "Elements of Matrix 2:" <<endl;

mat2.display();


    // Printing the elements of third matrix

cout<< "Elements of Matrix 3:" <<endl;

mat3.display();


        // Addition of two matrices using operator overloading

    mat4 = mat1 + mat3;

cout<< "Elements of Matrix after addition of Matrix 1 and Matrix 3:" <<endl;

mat4.display();


        // Equating two matrices using operator overloading

if (mat1 == mat2) {

cout<< "Matrix 1 is equals to Matrix 2" <<endl;

    }

else {

cout<< "Matrix 1 is not equals to Matrix 2" <<endl;

    }


        // Equating two matrices using operator overloading

if (mat1 == mat3) {

cout<< "Matrix 1 is equals to Matrix 3" <<endl;

    }

else {

cout<< "Matrix 1 is not equals to Matrix 3" <<endl;

    }


        return 0;

}
```

OUTPUT-



```
Elements of Matrix 1:
 [ 1, 2, ]
 [ 3, 4, ]

Elements of Matrix 2:
 [ 1, 2, ]
 [ 3, 4, ]

Elements of Matrix 3:
 [ 4, 3, ]
 [ 2, 1, ]

Elements of Matrix after addition of Matrix 1 and Matrix 3:
 [ 5, 5, ]
 [ 5, 5, ]

Matrix 1 is equals to Matrix 2
Matrix 1 is not equals to Matrix 3
```

The image part with relationship ID rId19 was not found in the file.

The image part with relationship ID rId19 was not found in the file.

EXPERIMENT NUMBER –Practical 6.4

STUDENT'S NAME – YASH RAJ

STUDENT'S UID – 21BCS11765

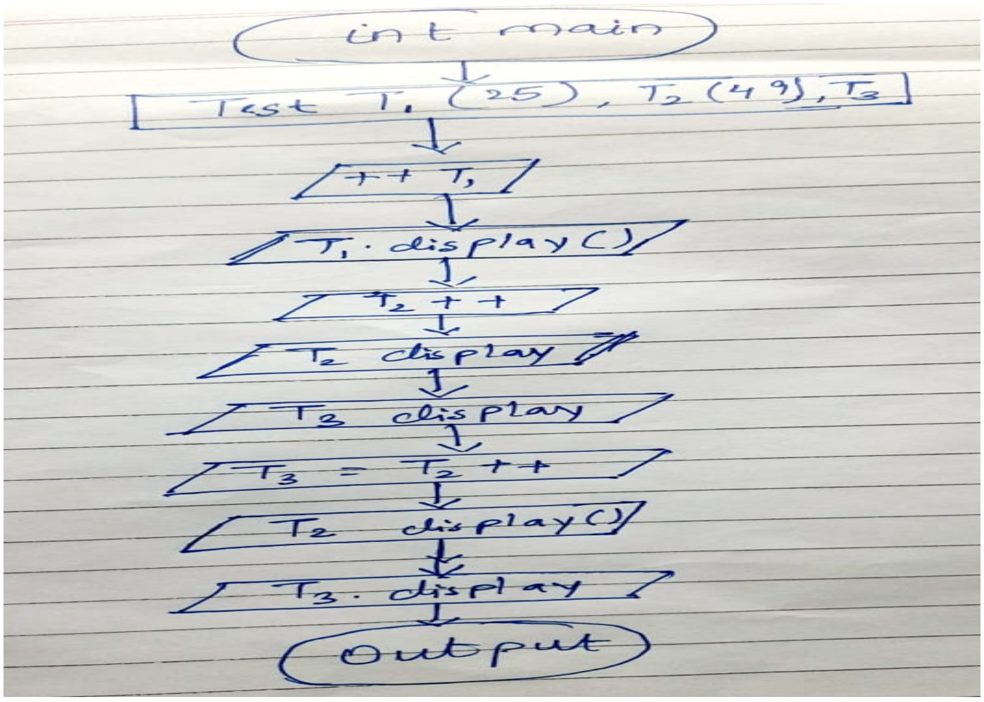
CLASS AND GROUP – 509-B

SEMESTER –2ND

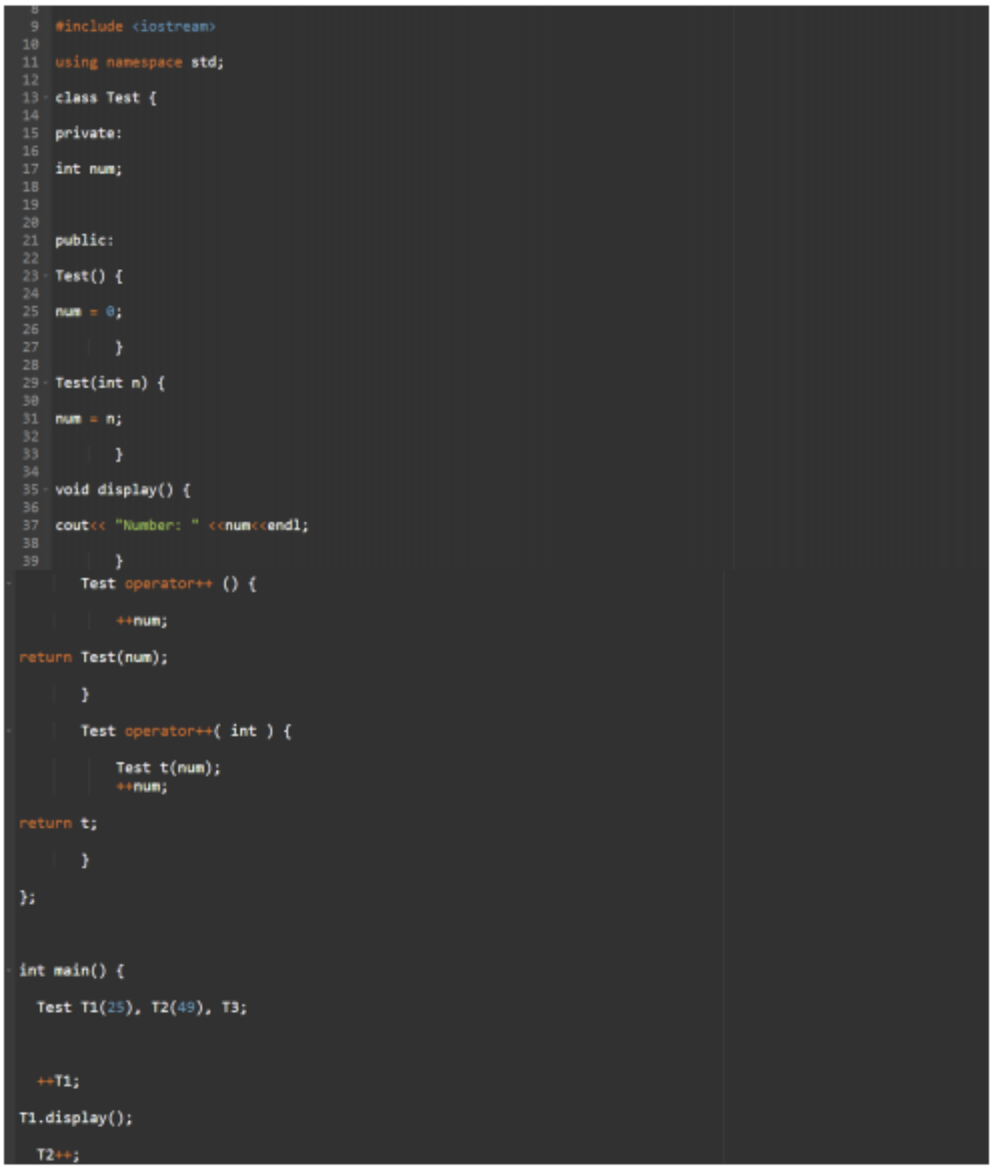
<u>TOPIC OF EXPERIMENT -</u> POLYMORPHISM

<u>AIM OF THE EXPERIMENT -</u> WAP to create a class Pairs. Objects of type Pairs can be used in any situation where ordered pairs are needed. Our Task is to overload operator >> and << so that objects of class Pairs are to be input and output in the form (5,3) (5,-6) (-5,6) or (-5,-3).There is no need to implement any constructor/method .

<u>FLOWCHART/ ALGORITHM-</u>

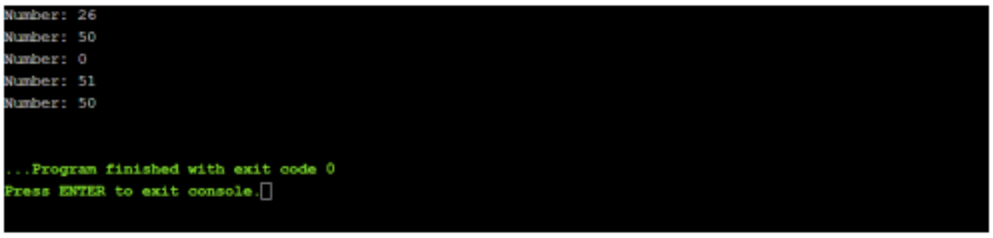


PROGRAM CODE-



```cpp
#include <iostream>

using namespace std;

class Test {

private:

int num;


public:

Test() {

num = 0;

        }

Test(int n) {

num = n;

        }

void display() {

cout<< "Number: " <<num<<endl;

        }
      Test operator++ () {

          ++num;

return Test(num);

      }

      Test operator++( int ) {

          Test t(num);
          ++num;

return t;

      }

};


int main() {

  Test T1(25), T2(49), T3;


  ++T1;

T1.display();

  T2++;
```

OUTPUT-



```
Number: 26
Number: 50
Number: 0
Number: 51
Number: 50


...Program finished with exit code 0
Press ENTER to exit console.
```

LEARNING OUTCOMES

- Identify situations where computational methods would be useful.
- Approach the programming tasks using techniques learnt and write pseudo-code.
- Choose the right data representation formats based on the requirements of the problem.
- Use the comparisons and limitations of the various programming constructs and choose the right one for the task.

EVALUATION COLUMN (To be filled by concerned faculty only)

Sr. No.	Parameters	Maximum Marks	Marks Obtained
1.	Worksheet Completion including writing learning objective/ Outcome	10	
2.	Post Lab Quiz Result	5	
3.	Student engagement in Simulation/ Performance/ Pre Lab Questions	5	
4.	Total Marks	20	