GR20 Regulations

II B.Tech I Semester

OBJECT ORIENTED PROGRAMMING
(GR20A2091)

B.Tech-Computer Science and Business System

GOKARAJU RANGARAJU
INSTITUTE OF ENGINEERING AND TECHNOLOGY
(Autonomous)

GOKARAJU RANGARAJU INSTITUTE OF ENGINEERING & TECHNOLOGY

OBJECT ORIENTED PROGRAMMING LAB

Course Code: GR20A2091 **L/T/P/C:0/0/4/2**

II Year I Semester

Course Objectives:

1. Reproduce basics of pointer handling, parameter passing mechanisms and also the concepts of object oriented programming as function overloading, in program.
2. Demonstrate the concepts related to class implementation
3. Construct programs for stack queue linked list considering access specifications
4. Test the concepts of operator overloading and templates using suitable programs
5. Design UML diagrams of Class, Sequence and Activity diagrams for any class concept considered.

Course Outcomes:

After the completion of the course, the student will be able to

1. Recall the concepts of Object oriented programming to solve real life problems
2. Demonstrate object oriented programming skills by using overloading, overriding, inheritance
   Concepts in developing solutions of a problem on hand.
3. Apply concepts of class hierarchy, templates and structure data using stacks and queue with
   help of OOP while developing programs.
4. Perceive and choose appropriate input-output formats and manipulators for developing
   interactive programs
5. Build systems with help of UML diagrams and OOPs concepts to solve real world problems.

TASK-1

1. a) Parameter passing: passing parameter by value vs by reference, passing array as constant pointer
   b) Function overloading: writing string operations like strcat and strncat, strcpy and strncpy as
   overloaded functions.
   c) Dynamically allocating space for a pointer depending on input and doing this repeatedly, depending on different inputs and finally de-allocating the pointer.

TASK-2

2. a) Define class complex with all possible operations: constructor, destructor, copy constructor, assignment operator with the data members stored as pointer to integers.
   b) Define class vector of integers with all possible operations like constructor, destructor, copy constructor and assignment operators
   c) Define class matrix of integers with all possible operations like constructor, destructor, copy
   constructor and assignment operators
   d) Define class matrix of integers using vector, with all possible operations like constructor,
   destructor, copy constructor and assignment operators

TASK-3

3. Define class stack, queue, linked-list, array, set using some data-type (int) with data members kept as private and functions kept in both protected and public sections.

TASK-4

4. a) Define class complex with all possible operators: constructor, destructor, copy constructor, assignment operator and operators >, <, >=, <=, ==, ++ (pre and post), +, +=, ( ), with the data members stored as pointer to integers.

   b) Define class vector of integers with all possible operations like constructor, destructor, copy constructor and assignment operators>, <, >=, <=, ==, ++ (pre and post), +, +=, ( )

TASK-5

5. a) Define class matrix of integers with all possible operations like constructor, destructor, copy constructor and assignment operators>, <, >=, <=, ==, ++ (pre and post), +, +=, ( ).

   b) Define class matrix of integers using vector, with all possible operations like constructor, destructor, copy constructor and assignment operators>, <, >=, <=, ==, ++ (pre and post), +, +=, ( ).

TASK-6

6. Define stack and queue inherited from array class, with standard functions and operators

TASK-7

7. a) Define a class called 'array' with data type passed as template type with constructor, destructor, copy constructor and assignment operators and index operator.

   b) Define template functions for compare and use it in the algorithms like bubble sort, insertion sort, merge sort.

TASK-8

8. Formatted input-output examples

TASK-9

9. Input manipulators

TASK-10

10. Overriding operators <<, >>

TASK-11

11. Define class model for complex number, student class, book class and show it using UML diagram as well as concrete class.

TASK-12

12. Show behavioural modelling through sequence diagram and activity diagram for workflow in a typical log-in, log-out situation.

Text Books:

1. The C++ Programming Language, Bjarne Stroustrup, Addison Wesley.
2. C++ and Object-Oriented Programming Paradigm, Debasish Jana, PHI Learning Pvt. Ltd.

Reference Books:

1. Programming – Principles and Practice Using C++, Bjarne Stroustrup, Addison Wesley.
2. The Design and Evolution of C++, Bjarne Stroustrup, Addison Wesley.

Index

TASK-1

TASK-1a) Parameter passing: passing parameter by value vs by reference, passing array as constant pointer

Aim: To Write a C++ Program for passing parameter by value vs by reference, passing array as constant pointer.

Program:

```
 // C++ program to swap two numbers using
// pass by value.
#include <iostream>
using namespace std;
void swap(int x, int y)
{
    int z = x;
    x = y;
    y = z;
}
  int main()
{
    int a = 45, b = 35;
    cout << "Before Swap\n";
    cout << "a = " << a << " b = " << b << "\n";

    swap(a, b);

    cout << "After Swap with pass by reference\n";
    cout << "a = " << a << " b = " << b << "\n";
}
```

Expected Output:

```
Before Swap
a = 45 b = 35
After Swap with pass by reference
a = 45 b = 35
```

```
// C++ program to swap two numbers using
// pass by reference.
#include <iostream>
```

```
using namespace std;
void swap(int& x, int& y)
{
   int z = x;
   x = y;
   y = z;
}
int main()
{
   int a = 45, b = 35;
   cout << "Before Swap\n";
   cout << "a = " << a << " b = " << b << "\n";

   swap(a, b);

   cout << "After Swap with pass by reference\n";
   cout << "a = " << a << " b = " << b << "\n";
}
```

Expected Output:

```
        Before Swap
   a = 45 b = 35
   After Swap with pass by reference
   a = 35 b = 45
```

Passing array as constant pointer

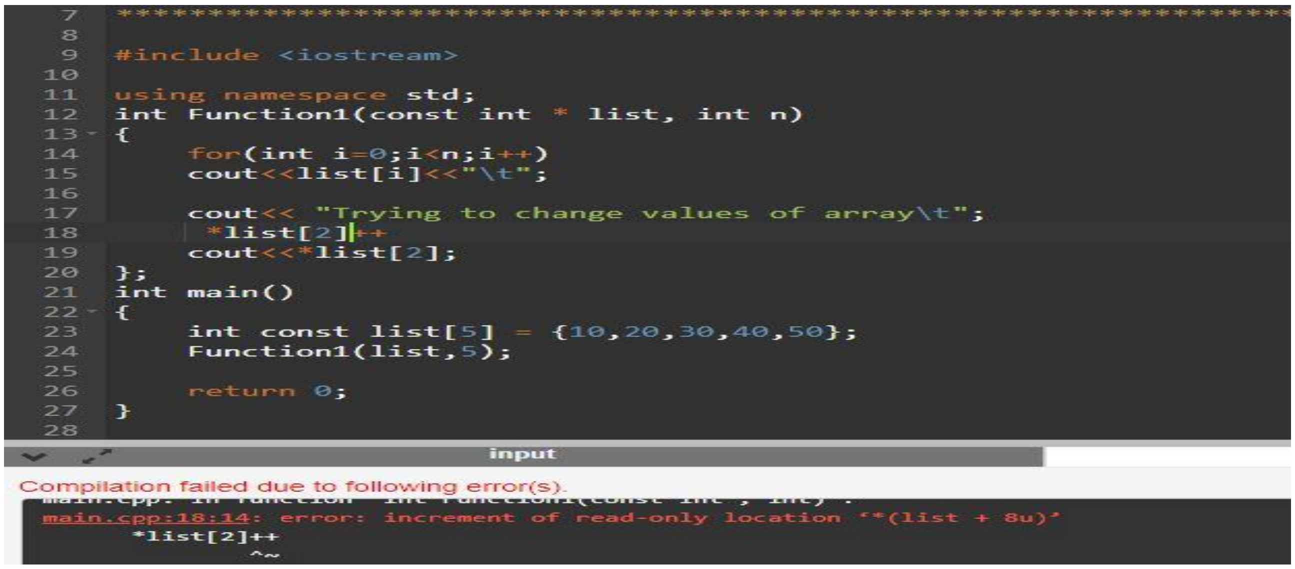
Example:

```
int Function1(const int * list);  // the target of list can't
                                  // be changed in the function
```

```
//C++ program for passing array as constant pointer
#include <iostream>

using namespace std;
int Function1(const int * list, int n)
{
   for(int i=0;i<n;i++)
   cout<<list[i]<<"\t";

   cout<< "Trying to change values of array\t";
    *list[2]++
   cout<<*list[2];
};
int main()
```

```
{
   int const list[5] = {10,20,30,40,50};
   Function1(list,5);

   return 0;
}
```

Expected Output:



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

#include <iostream>

using namespace std;
int Function1(const int * list, int n)
{
      for(int i=0;i<n;i++)
      cout<<list[i]<<"\t";

      cout<< "Trying to change values of array\t";
       *list[2]++
      cout<<*list[2];
};
int main()
{
      int const list[5] = {10,20,30,40,50};
      Function1(list,5);

      return 0;
}
```

input

Compilation failed due to following error(s).

```
main.cpp: In function 'int Function1(const int *, int)':
main.cpp:18:14: error: increment of read-only location '*(list + 8u)'
     *list[2]++
           ^~
```

Task-1b): Function overloading: writing string operations like strcat and strncat, strcpy and strncpy as overloaded functions.

Aim: To Write a C++ Program for Function overloading: writing string operations like strcat and strncat, strcpy and strncpy as overloaded functions.

Program:

```
#include<iostream>
#include<cstring>
using namespace std;
void conc(char *, char *);
void conc(char *,  char *, int);
void copy(char *, char *);
void copy(char *, char *, int);

void copy(char *source, char *destn)
{

   destn=source;
```

```
    cout<<"After copy\n";
    cout<<source<<"\t"<<destn<<"\n";
}

void copy(char *source, char *destn, int n)
{
    int i;

    for(i=0;i<n; i++)
        destn[i] = source[i];

    cout<<"After copy\n";
    cout<<source<<"\t"<<destn<<"\n";

}

void conc(char *x,char *y)
{
int p;
        for(p=0; x[p] != '\0'; p++);//pointing to the index of the last character of x

    for(int q=0; y[q] != '\0'; q++,p++)
    {
        x[p]=y[q];
    }

    x[p]='\0';
    cout<<"Concatenated String:\n";
    cout<<x<<endl;
}

 void conc(char *dest, char *source, int n)
{
int p;
        for(p=0; dest[p] != '\0'; p++);//pointing to the index of the last character of destination

    for(int q=0; q<n; q++,p++)
    {
        dest[p]=source[q];
    }

    dest[p]='\0';
    cout<<"Concatenated String:\n";
    cout<<dest<<endl;
}
```

```
int main()
{
    int n;
    char x[100], y[100],z[100];
    cout<<"String 1:\n";
    cin>>x;

    cout<<"String 2:\n";
    cin>>y;
     conc(x,y);

     cout<<"Enter the source string for N character concatanation";
     cin>>z;
     cout<<"String concatanation of N characters";
     cout<<"Enter N";
     cin>>n;
      conc(x,z,n);

     cout<<"Enter strings to be copied Source & Destination\n";
     cin>>x;
     cin>>y;
        copy(x,y);
        cout<<"Enter number of characters to be copied from source\n";
        cin>>n;
        copy(x,y,n);


    return 0;
}
```

Expected Output:

```
String 1:
Gokaraju
String 2:
Rangaraju
Concatenated String:
GokarajuRangaraju
Enter the source string for N character concatanationInstitute
String concatanation of N charactersEnter N5
Concatenated String:
GokarajuRangarajuInsti
Enter strings to be copied Source & Destination
college
Institute
After copy
```

```
college college
Enter number of characters to be copied from source
4
After copy
college collitute
```

Task 1c): Dynamically allocating space for a pointer depending on input and doing this repeatedly, depending on different inputs and finally de-allocating the pointer.

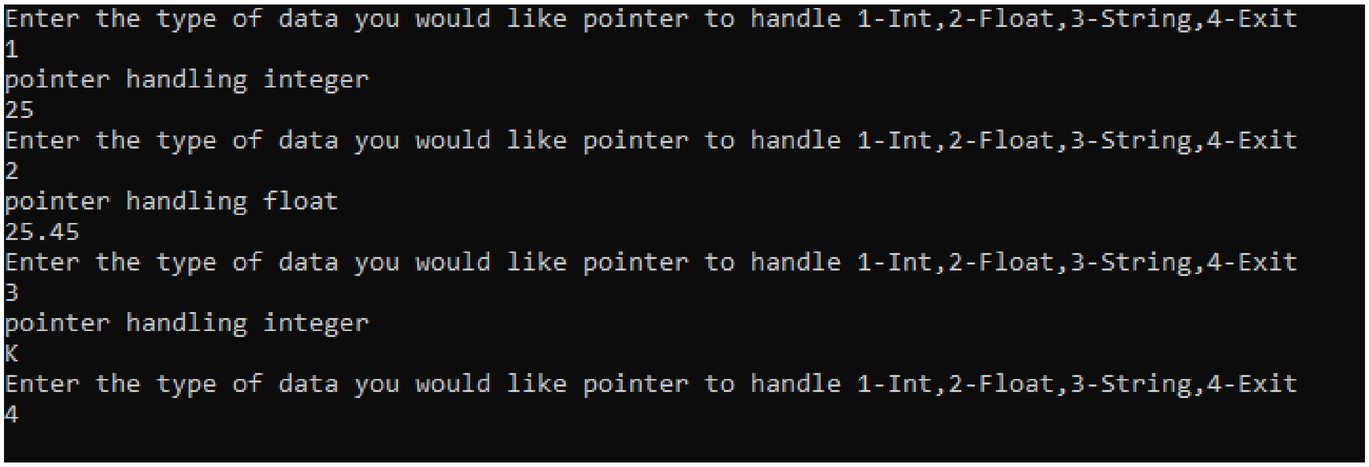
Aim: Write a C++ Program for Dynamically allocating space for a pointer depending on input and doing this repeatedly, depending on different inputs and finally de-allocating the pointer.

Program:

```
#include<iostream>
#include<cstring>
using namespace std;
int main()
{  int choice;
   void *ptr;

            while(1)
   {
      cout<< "Enter the type of data you would like pointer to handle \t 1-Int,\t 2-Float,\t 3-String,\t
4-Exit\n";
   cin>> choice;
   switch (choice)
   {
             case 1: { int (*ptr) = new(int);
              *ptr = 25;
              cout<<"pointer handling integer\n";
              cout<<*ptr<<endl;
              break; }
       case 2: {float (*ptr) = new(float);
              *ptr = 25.45;
              cout<<"pointer handling float\n";
              cout<<*ptr<<endl;
              break;
           }
```

```
            case 3:{ char (*ptr) = new(char);
                    *ptr = 'K';
                    cout<<"pointer handling integer\n";
                    cout<<*ptr<<endl;
                    break;}
            case 4: exit(0);
        }

        delete(ptr);
        }
    }
```

Output:



```
Enter the type of data you would like pointer to handle 1-Int,2-Float,3-String,4-Exit
1
pointer handling integer
25
Enter the type of data you would like pointer to handle 1-Int,2-Float,3-String,4-Exit
2
pointer handling float
25.45
Enter the type of data you would like pointer to handle 1-Int,2-Float,3-String,4-Exit
3
pointer handling integer
K
Enter the type of data you would like pointer to handle 1-Int,2-Float,3-String,4-Exit
4
```

TASK-2

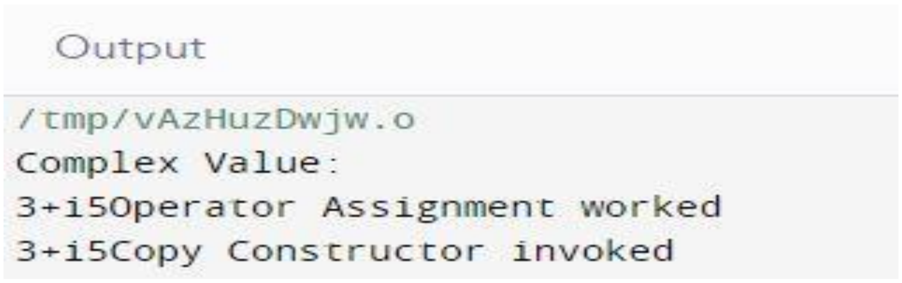
Task 2a): Define class complex with all possible operations: constructor, destructor, copy constructor, assignment operator with the data members stored as pointer to integers.

Aim: Write a C++ Program for Define class complex with all possible operations: constructor, destructor, copy constructor, assignment operator with the data members stored as pointer to integers.

Program:

```
#include <iostream>
using namespace std;
class Complex{
    private:
      int *real;
      int *imag;
    public:
       Complex()
       {
          real = new int(sizeof(int));
          imag =new int(sizeof(int));
       }
       Complex (int x, int y)
       {
          real = new int(sizeof(int));
          *real=x;
          imag =new int(sizeof(int));
          *imag = y;
       }
       Complex(const Complex &c)  //Copy Constructor
       {
          cout<<"Copy Constructor invoked\n";
          *real= *c.real;
         *imag=*c.imag;

       }
       ~Complex()
       {
         free(real);
         free(imag);

       }
       void display()
```

```
        {
           cout<<"Complex Value:\n"<<*real<<"+i"<<*imag;
        }
        void read()
        { int i,r;
           cout<< "enter Real Value\n";
           cin>>r;
           cout<<"enter Imaginary Value\n";
           cin>>i;
           *real=r;
           *imag=i;
        }
        int getreal()
        {
           return *real;
        }
        int getimag()
        {
           return *imag;
        }
        void operator =(const Complex &c) //Assignment Operator
        {
           *this->real= *c.real;
           *this->imag= *c.imag;
           cout<<"Operator Assignment worked\n";
           cout<<*this->real<<"+i"<<*this->imag;
        }

  };

  int main()
  {
     Complex c1;
     Complex c2(3,5);
     //c1.read();
     //c1.display();
     c2.display();
     c1 = c2; //assignment operator
     Complex c3=c1;
     c3.display();
     return 0;
    }
```
Output:


Output

```
/tmp/vAzHuzDwjw.o
Complex Value:
3+i5Operator Assignment worked
3+i5Copy Constructor invoked
```

Task 2b): Define class vector of integers with all possible operations like constructor, destructor, copy constructor and assignment operators

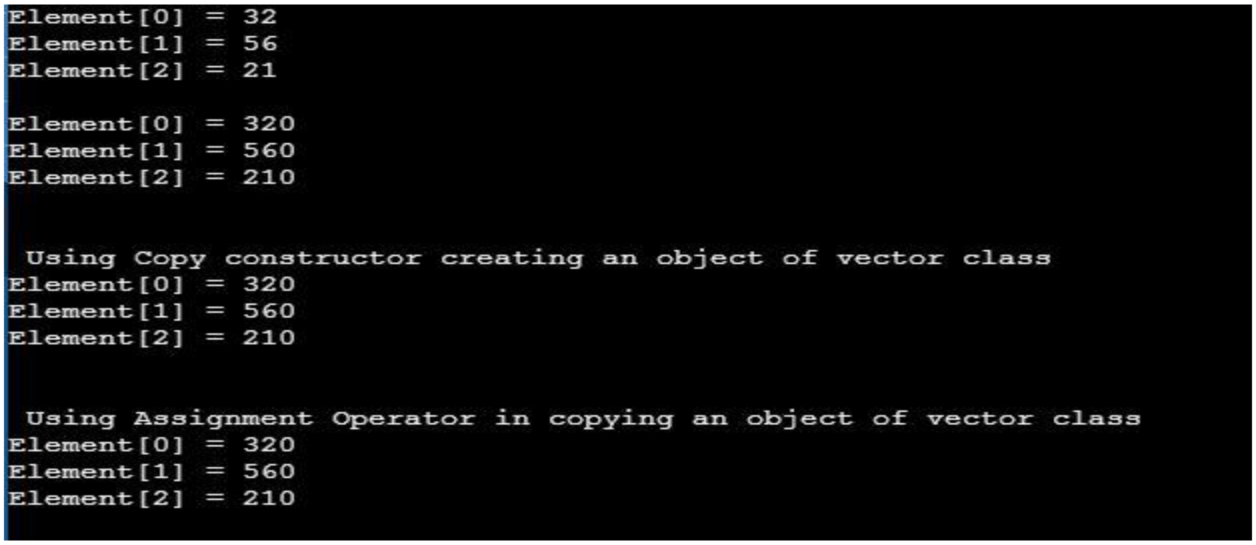
Aim: Write a C++ Program for Defining class vector of integers with all possible operations like constructor, destructor, copy constructor and assignment operators

Program:

```
 #include <iostream>
#include <vector>
using namespace std;
class Object
{
   private:
                    std::vector<int> m_VectorOfInts;
          public:
                    Object()
                    {

                    }
                    ~Object()
                    {

                    }
                    Object(const Object  &O1)
                    {

                              m_VectorOfInts=O1.m_VectorOfInts;

                    }

                    void operator =(Object const &O1)
                    {
                       for( unsigned int i = 0; i < O1.m_VectorOfInts.size(); i++ )
                         {
                             m_VectorOfInts[i]=O1.m_VectorOfInts[i];
```

```
                        }
                }


                void AddInt(int num)
                {
                        m_VectorOfInts.push_back(num);
                }
                std::vector<int> GetCopyOfVector()
                {
                        return m_VectorOfInts;
                }
                void DisplayVectorContents()
                {
                        for( unsigned int i = 0; i < m_VectorOfInts.size(); i++ )
                        {
                                std::cout << "Element[" << i << "] = " << m_VectorOfInts[i] <<
                                std::endl;
                        }
                        std::cout << std::endl;
                }
};
int main()
{
   int choice;
        // Create our class an add a few ints
        Object obj,obj1;
        obj.AddInt(32);
        obj.AddInt(56);
        obj.AddInt(21);
   obj1.AddInt(320);
        obj1.AddInt(560);
        obj1.AddInt(210);
        // Display the vector contents so far
        obj.DisplayVectorContents();
   obj1.DisplayVectorContents();
  cout<<"\n Using Copy constructor creating an object of vector class\n";
Object objCopyCon(obj1);
 objCopyCon.DisplayVectorContents();
    cout<<"\n Using Assignment Operator in copying an object of vector class\n";
    obj=objCopyCon;
  obj.DisplayVectorContents();
        return 0;
}
```

output:


```
Element[0] = 32
Element[1] = 56
Element[2] = 21

Element[0] = 320
Element[1] = 560
Element[2] = 210


 Using Copy constructor creating an object of vector class
Element[0] = 320
Element[1] = 560
Element[2] = 210


 Using Assignment Operator in copying an object of vector class
Element[0] = 320
Element[1] = 560
Element[2] = 210
```

Task 2c): Define class matrix of integers with all possible operations like constructor, destructor, copy constructor and assignment operators

Aim: Define class matrix of integers with all possible operations like constructor, destructor, copy constructor and assignment operators

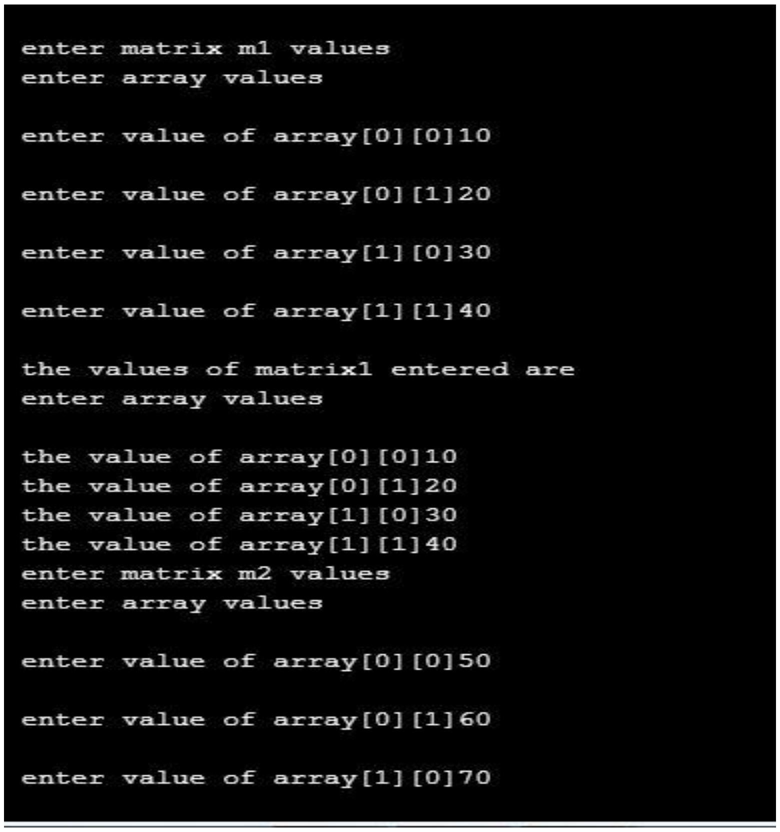
Program:

```
#include<iostream>
using namespace std;
#include<cstdlib>
#include<new>
class matrix
{
 int **array;
 int row;
 int column;
 public:
  //default constructor
  matrix()
  {
   array=NULL;
   row=0;
   column=0;
  }
  //single parameter constructor(square matrix)
  matrix(int length)
  {
```

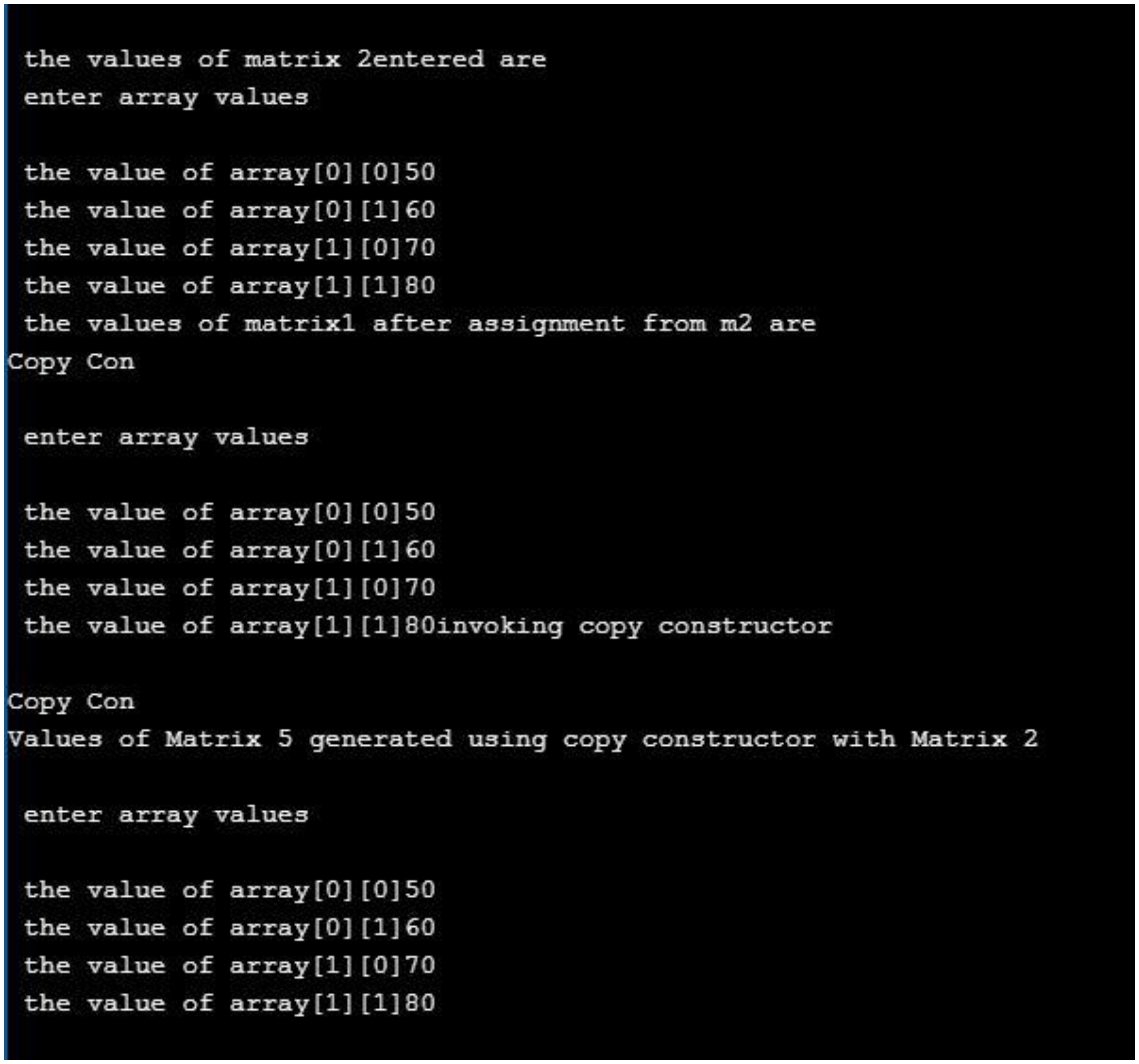
```
  int i,j;
  array=(int **)new int[length];
  row=length;
  column=length;
  for(i=0;i<length;i++)
   array[i]=new int[length];
  for(i=0;i<length;i++)
   for(j=0;j<length;j++)
    array[i][j]=0;
 }
 //double parameterized constructor(rectangular matrix)
 matrix(int r,int c)
 {
  int i,j;
  row=r;
  column=c;
  array=(int **)new int[r];
  for(i=0;i<r;i++)
   array[i]=new int[c];

  for(i=0;i<r;i++)
   for(j=0;j<c;j++)
   array[i][j]=0;
 }
 //copy constructor
 matrix(const matrix &c)
 {
  int i,j;
  cout<<"\nCopy Con\n";
  array=(int **)new int[c.row];
  for(i=0;i<c.row;i++)
   array[i]=new int[c.column];
  row=c.row;
  column=c.column;
  for(i=0;i<c.row;i++)
   for(j=0;j<c.column;j++)
    array[i][j]= c.array[i][j];
 }
//destructor
 ~matrix()
 {
  delete []array;
 }
//Assignment Operator
matrix operator=(const matrix &mat)
{
 int i,j;

 if(row!=mat.row || column!=mat.column)
 {
  cout<<"\n\nAssignment of different matrices is not possible..\n";
  exit(1);
 }
 for(i=0;i<row;i++)
  for(j=0;j<column;j++)
   array[i][j]=mat.array[i][j];
 return (*this);
 }
 void  operator delete[](void *pointer)
 {
  free(pointer);
 }
friend istream& operator>>(istream &cin,matrix &c);
friend ostream& operator<<(ostream &cout,matrix &c);
};
istream& operator>>(istream &cin,matrix &c)
{
 int i,j;
 cout<<"\n enter array values"<<endl;
 for(i=0;i<c.row;i++)
 {
  for(j=0;j<c.column;j++)
  {
   cout<<"\n enter value of array["<<i<<"]["<<j<<"]";
   cin>>c.array[i][j];
  }
 }
 return cin;
}
ostream& operator<<(ostream &cout,matrix &c)
{
 int i,j;
 cout<<"\n enter array values"<<endl;
 for(i=0;i<c.row;i++)
 {
  for(j=0;j<c.column;j++)
  {
   cout<<"\n the value of array["<<i<<"]["<<j<<"]";
   cout<<c.array[i][j];
  }
 }
 return cout;
```

```
}
int main()
{
 matrix m1(2),m2(2),m3(2),m5(2);
 cout<<"\n enter matrix m1 values";
 cin>>m1;
 cout<<"\n the values of matrix1 entered are";
 cout<<m1;
 cout<<"\n enter matrix m2 values";
 cin>>m2;
 //cout<<m2;
 //cin>>m3;
 cout<<"\n the values of matrix 2entered are";
 cout<<m2;
 //m3=m1+m2;
 cout<<"\n the values of matrix1 after assignment from m2 are";
 m1 = m2; cout<<m1;
 cout<<"invoking copy constructor\n";
 m5 = m2;
 cout<<"Values of Matrix 5 generated using copy constructor with Matrix 2\n";
 cout<<m5;
}
```

OutPut:



```
enter matrix m1 values
enter array values

enter value of array[0][0]10

enter value of array[0][1]20

enter value of array[1][0]30

enter value of array[1][1]40

the values of matrix1 entered are
enter array values

the value of array[0][0]10
the value of array[0][1]20
the value of array[1][0]30
the value of array[1][1]40
enter matrix m2 values
enter array values

enter value of array[0][0]50

enter value of array[0][1]60

enter value of array[1][0]70
```



```
 the values of matrix 2entered are
 enter array values

 the value of array[0][0]50
 the value of array[0][1]60
 the value of array[1][0]70
 the value of array[1][1]80
 the values of matrix1 after assignment from m2 are
Copy Con

 enter array values

 the value of array[0][0]50
 the value of array[0][1]60
 the value of array[1][0]70
 the value of array[1][1]80invoking copy constructor

Copy Con
Values of Matrix 5 generated using copy constructor with Matrix 2

 enter array values

 the value of array[0][0]50
 the value of array[0][1]60
 the value of array[1][0]70
 the value of array[1][1]80
```

Task 2d): Define class matrix of integers using vector, with all possible operations like constructor, destructor, copy constructor and assignment operators

Aim: Write a C++ Program for Define class matrix of integers using vector, with all possible operations like constructor, destructor, copy constructor and assignment operators

Program:

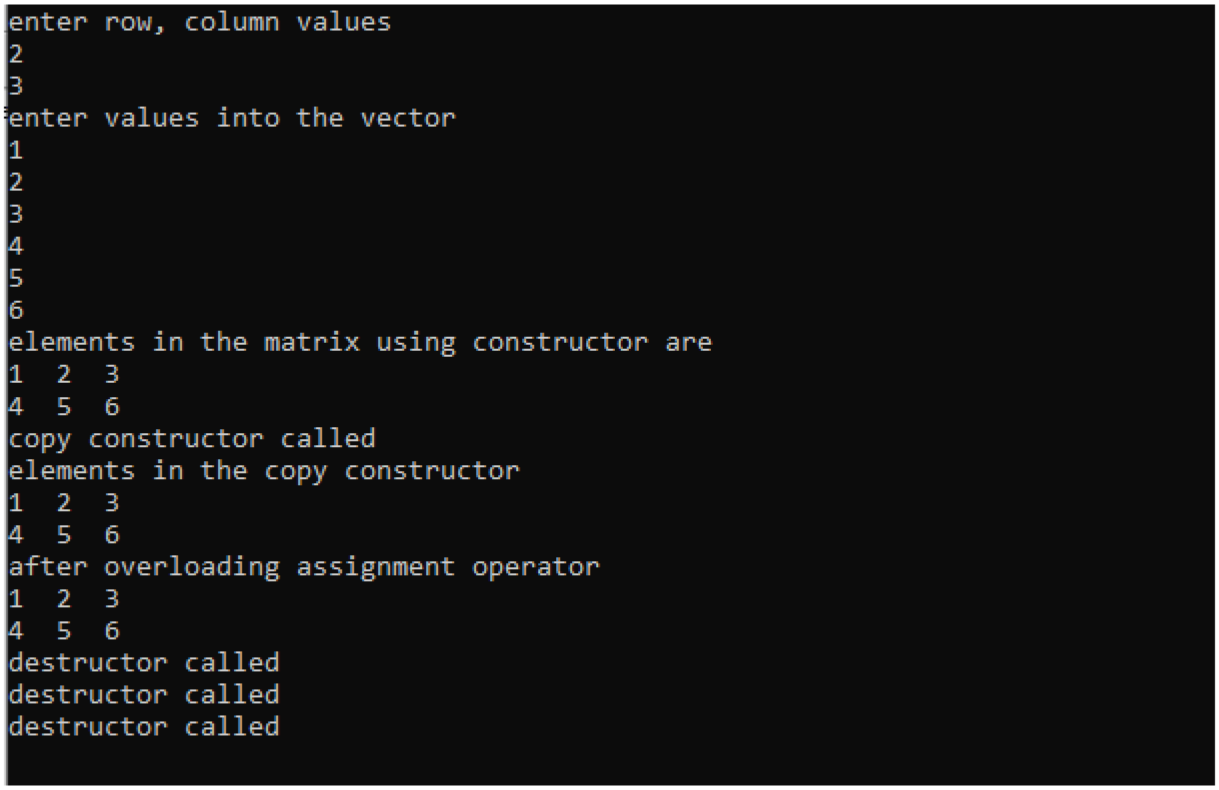
```
#include<iostream>
#include<vector>

using namespace std;
```

```
class Matrix
{
  int x,y;                  // redundant class data erased
  vector< vector <int> > v;

  public:
  Matrix(): x(0), y(0), v(0)
  {
  }

  Matrix(int m, int n) // Main constructor
  {
     x=m;
     y=n;
     v = vector< vector<int> >(x,vector <int>(y, 0));   // in original version, this class data was redefined
locally
     cout<< "enter values into the vector" << endl;
     for(int i = 0; i < x; ++i)
     {
        for (int j = 0; j < y; ++j)
        {
           cin >> v[i][j];
        }
     }
  }

  Matrix(const Matrix &t)
  {
     x=t.x;
     y=t.y;
     v=t.v;
     cout << "copy constructor called" << endl;
  }

  ~Matrix()
  {
     cout << "destructor called" << endl;
  }

  Matrix& operator=(const Matrix &r)   // needs to output a reference
  {
     if (this == &r)   // check self-assignment
        return *this;

     x = r.x;
```

```
        y = r.y;
        vector< vector<int> > b(x,vector <int>(y, 0));
        for (int i = 0; i < x; ++i)
        {
            for (int j = 0; j < y; ++j)
            {
                b[i][j] = r.v[i][j];
            }
        }

        for (int i = 0; i < x; i++)
        {
            for (int j = 0; j < y; j++)
            {
                cout << b[i][j] << "  ";
            }
            cout << endl;
        }

        return *this;
    }

    void print()
    {
        for (int i = 0; i < x; i++)
        {
            for (int j = 0; j < y; j++)
            {
                cout << v[i][j] << "  ";
            }
            cout << endl;
        }
    }
};

int main()
{
    int d1,d2;
    cout << "enter row, column values" << endl;
    cin >> d1 >> d2;
    Matrix ob1();
    Matrix ob2(d1,d2);
    cout << "elements in the matrix using constructor are" << endl;
    ob2.print();
    Matrix ob3(ob2);
    cout << "elements in the copy constructor" << endl;
```

```
  ob3.print();
  cout << "after overloading assignment operator" << endl;
  Matrix ob4;
  ob4=ob2;

  return 0;
}
```

Output:



```
enter row, column values
2
3
enter values into the vector
1
2
3
4
5
6
elements in the matrix using constructor are
1  2  3
4  5  6
copy constructor called
elements in the copy constructor
1  2  3
4  5  6
after overloading assignment operator
1  2  3
4  5  6
destructor called
destructor called
destructor called
```

TASK 3

TASK 3: Define class stack, queue, linked-list, array, set using some data-type (int) with data members kept as private and functions kept in both protected and public sections.

Aim: Write a C++ Program for Define class stack, queue, linked-list, array, set using some data-type (int) with data members kept as private and functions kept in both protected and public sections.

Program:

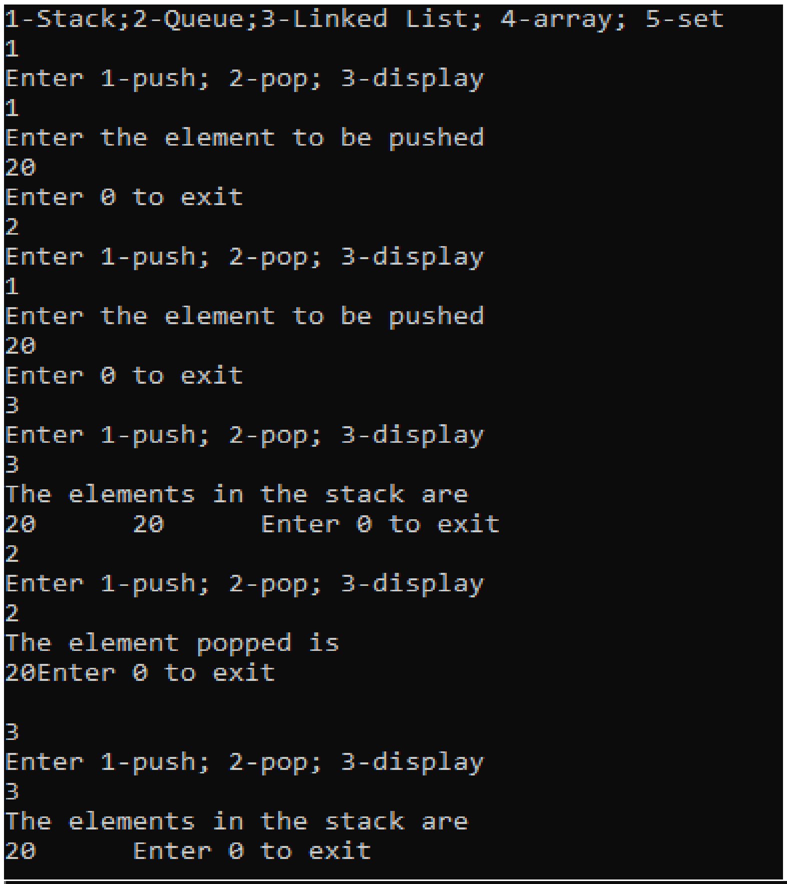
```
#include<iostream>
#include<set>
using namespace std;
class setimp{
  private:
        // Set with values
    set<int> s1;
    protected:
    int searchele(int val)
    {
      // Check if the iterator returned is not the ending of set
    if(s1.find(val) != s1.end())
        cout<<"The set contains "<<val<<endl;
    else
        cout<<"The set does not contains "<<val<<endl;
    }
     void removeele()
     {
        int ele;
        // Erasing element value = 1
    cout<<"\nEnter a specific element you want to remove from set\n";
    cin>>ele;
    s1.erase(ele);

    cout<<"Valuesafter deleting\n";
    display();
    // Erasing the first element
    s1.erase(s1.begin());
    cout<<"Valuesafter deleting first element\n";
    display();
     }
    public:
    setimp()
    {  int n,ele,i;
        cout<<"Enter number of values you want to inseert\n";
        cin>>n;
        for(i=0;i<n;i++)
        {
            cout<<"\nEnter element\n";
            cin>>ele;
            s1.insert(ele);
        }
    }
    void display()
    {
        // Iterator for the set
    set<int> :: iterator it;
    // Print the elements of the set
    for(it=s1.begin(); it != s1.end();it++)
        cout<<*it<<" ";
    cout<<endl;
    }
};
class setuser: public setimp{
    public:
     void eraseele()
     {
         cout<<"You choose to remove element from Set\n";
         removeele();
     }
    void search()
     {  int val;
         cout<<"Enter the element to be checked in Set\n";
         cin>>val;
         searchele(val);
     }
 };
 //Linked List Implementation
class Node
{
private:
    int info;
    Node* next;
    Node *first,*last;
protected:
       void create();
    void insert();
    void delet();
    void display();
    void search();
public:
    Node()
    {
        first=NULL;
        last=NULL;
    }
};
class List:public Node
{
 public:
   void createit()
    {
        create();
    }
    void insertit()
    {
        insert();
    }
    void deleteit()
    {
        delet();
    }
    void displaylist()
    {
        display();
    }
    void searchelement()
    {
        search();
    }
    void createlist()
    {
        create();
    }
};
void Node::create()
{
    Node *temp;
    temp=new Node;
    int n;
    cout<<"\nEnter an Element:";
    cin>>n;
    temp->info=n;
    temp->next=NULL;
    if(first==NULL)
    {
        first=temp;
        last=first;
    }

    else
    {
        last->next=temp;
        last=temp;
    }
}
void Node::insert()
{
    Node *prev,*cur;
    prev=NULL;
    cur=first;

    int count=1,pos,ch,n;
    Node *temp=new Node;
    cout<<"\nEnter an Element:";
    cin>>n;
    temp->info=n;
    temp->next=NULL;

    cout<<"\nINSERT AS\n1:FIRSTNODE\n2:LASTNODE\n3:IN BETWEEN FIRST&LAST NODES";
    cout<<"\nEnter Your Choice:";
    cin>>ch;
    switch(ch)
    {
    case 1:
        temp->next=first;
        first=temp;

        break;
    case 2:
        last->next=temp;
        last=temp;

        break;
    case 3:
        cout<<"\nEnter the Position to Insert:";
        cin>>pos;
        while(count!=pos)
        {
            prev=cur;
            cur=cur->next;
            count++;
        }
        if(count==pos)
        {
            prev->next=temp;
            temp->next=cur;
        }
        else
            cout<<"\nNot Able to Insert";
        break;

    }

}
void Node::delet()
{
    Node *prev=NULL,*cur=first;
    int count=1,pos,ch;
    cout<<"\nDELETE\n1:FIRSTNODE\n2:LASTNODE\n3:IN BETWEEN FIRST&LAST NODES";
    cout<<"\nEnter Your Choice:";
    cin>>ch;
    switch(ch)
    {
    case 1:
        if(first!=NULL)
        {
            cout<<"\nDeleted Element is "<<first->info;
            first=first->next;
        }
        else
            cout<<"\nNot Able to Delete";
        break;
    case 2:
        while(cur!=last)
        {
            prev=cur;
            cur=cur->next;
        }
        if(cur==last)
        {
            cout<<"\nDeleted Element is: "<<cur->info;
            prev->next=NULL;
            last=prev;
        }
        else
            cout<<"\nNot Able to Delete";
        break;
    case 3:
        cout<<"\nEnter the Position of Deletion:";
        cin>>pos;
        while(count!=pos)
        {
            prev=cur;
            cur=cur->next;
            count++;
        }
        if(count==pos)
        {
            cout<<"\nDeleted Element is: "<<cur->info;
            prev->next=cur->next;
        }
        else
            cout<<"\nNot Able to Delete";
        break;
    }
}
void Node::display()
{
    Node *temp=first;
    if(temp==NULL)
    {
        cout<<"\nList is Empty";
    }
    while(temp!=NULL)
    {
        cout<<temp->info;
        cout<<"-->";
        temp=temp->next;
    }
    cout<<"NULL";
}
void Node::search()
{
    int value,pos=0;
    bool flag=false;
    if(first==NULL)
    {
        cout<<"List is Empty";
        return;
    }
    cout<<"Enter the Value to be Searched:";
    cin>>value;
    Node *temp;
    temp=first;
    while(temp!=NULL)
    {
        pos++;
        if(temp->info==value)
        {
            flag=true;
            cout<<"Element"<<value<<"is Found at "<<pos<<" Position";
            return;
        }
        temp=temp->next;
    }
    if(!flag)
    {
        cout<<"Element "<<value<<" not Found in the List";
    }
}


//Queue Implementation.
class queue
{
        private :

                int *arr ;
                int front, rear ;
                int MAX ;
        protected:
                void addq ( int item ) ;
                int delq( ) ;
        public :
                queue( int maxsize = 10 ) ;

} ;

queue :: queue( int maxsize )
{
        MAX = maxsize ;
        arr = new int [ MAX ];
        front = -1 ;
        rear = -1 ;
}

void queue :: addq ( int item )
{
        if ( rear == MAX - 1 )
        {
                cout << "\nQueue is full" ;
                return ;
        }
        rear++ ;
        arr[rear] = item ;
        if ( front == -1 )
                front = 0 ;
        if (front == - 1)
   cout<<"Queue is empty"<<endl;
   else {
      cout<<"Queue elements are : ";
      for (int i = front; i <= rear; i++)
      cout<<arr[i]<<" ";
         cout<<endl;
   }
}
int queue :: delq( )
{
        int data ;

        if ( front == -1 )
        {
                cout << "\nQueue is Empty" ;
                return -1000 ;
        }

        data = arr[front] ;
        arr[front] = 0 ;
        if ( front == rear )
                front = rear = -1 ;
        else
                front++ ;
            if (front == - 1)
   cout<<"Queue is empty"<<endl;
   else {
      cout<<"Queue elements are : ";
      for (int i = front; i <= rear; i++)
      cout<<arr[i]<<" ";
         cout<<endl;
   }
        return  data ;
}

class Q: public queue
```

```
{
    public:
       void insert(int ele)
       {
           addq(ele);
       }

       int remov()
       {
           int x;
           x=delq();
           return x;
       }
};

class STACK{
  int arr[5],top;
  protected:
  void topinc()
  {
     top++;
  }
  void topdec()
  {
     top--;
  }
  int gettop()
  {
     return top;
  }
  void setvalue(int e)
  {
     arr[top]= e;
  }
  int delval()
  {
     return arr[top];
  }
  public:
  STACK()
  {
     int i;
     for(i=0;i<5;i++)
       arr[i]=-1;

       top=-1;
  }
  ~STACK()
  {
      cout<<"\n STACK destructor working\n";
  }

  void display()
  {
      int temp=top;
      if(temp!=-1)
      for(;temp>=0;temp--)
       cout<<arr[temp]<<"\t";

      else
        cout<<"No elements to display\n";
  }
};

class stk:public STACK
{ public:
  void push(int ele)
  {
      if(gettop()==4)
       cout<<"\nStack Overflow\n";
      else
      {
          topinc();
          setvalue(ele);
      }
  }
   int pop()
   { int e;
       if(gettop()==-1)
         {cout<<"Stack Underlow\n";
           return -1;
         }
        else
        {
            e=delval();
            topdec();
            return e;
        }
   }
};
class array{
private:
```

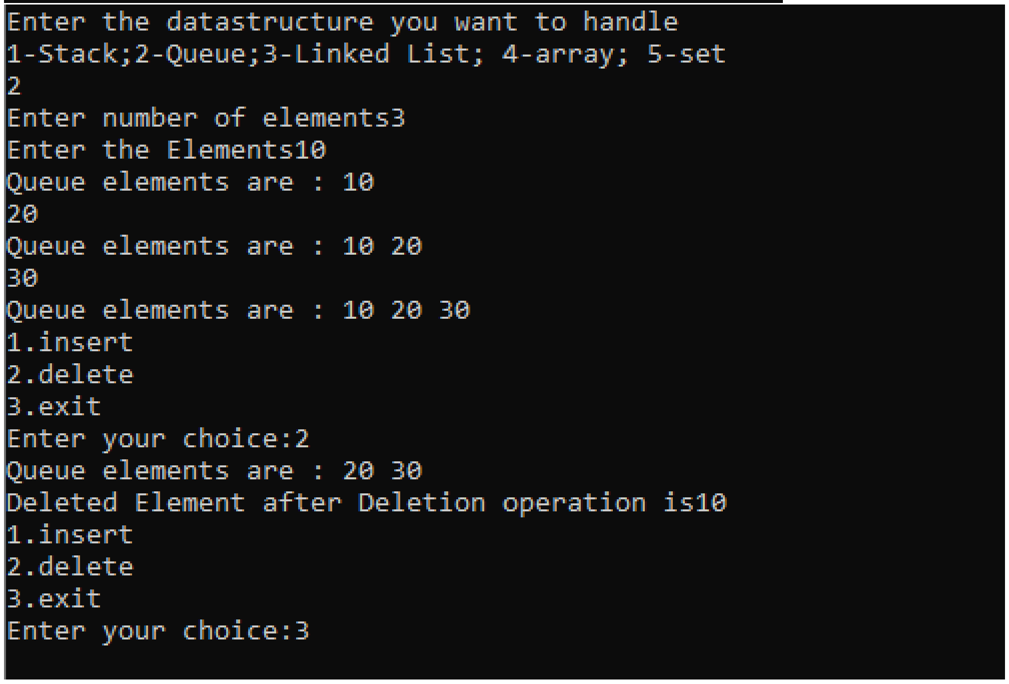
```
            int a[50],loc,item,n;
public:
    array(){
       loc=0;item=0;n=0;

    }

            void init();
            void disp(); //initial data assignmentvoid traverse(); //process is display (assumed)void insert();
   void insert();
    void del();
    void search();
};
void array :: init(){
            cout<<"Enter the size of an array";
       cin>>n;
    cout<<"Enter the elements into erray"<<endl;
               for(int i=0;i<n;i++)
                        cin>>a[i];
            cout<<endl;
            }
void array :: disp(){
     cout<<"The elements in array are:"<<endl;
     for(int i=0;i<n;i++)
     cout<<a[i]<<" ";
            cout<<endl;
}

/*All Operations Except Merging are  performed on arrA*/
void array :: insert(){

     int i;
     cout<<"\n\n*****Inserting Element*****\n";
     cout<<"\nEnter Location of insertion : ";
     cin>>loc;
     loc--;
     if(loc<0 || loc>=n)
     {
    cout<<"\n\nInvalid Position\n";
    exit(0);
     }
     cout<<"Enter Item value to be inserted : ";
     cin>>item;
     for(i=n-1;i>=loc;i--){
     a[i+1] = a[i];
     }
     a[loc]=item;
     n++;
     cout<<"\nItem is Inserted\n";


}

void array :: del(){

     int i;
     cout<<"\n\n*****Deleting Element*****\n";
     if(n < 0){
    cout<<"\nArray is Empty\nDeletion Not Possible\n";
   exit(0);
      }
     cout<<"\nEnter Location of deletion : ";
     cin>>loc;
     loc--;
     if(loc<0 || loc>=n)
     {
    cout<<"\n\nInvalid Position\n";
    exit(0);
     }
     cout<<"\nItem deleted is : "<<a[loc];
     for(i=loc;i<n;i++){
     a[i] = a[i+1];
     }
     a[n-1]=0;
     n--;

}

void array :: search(){

     int i,found=-1;
     cout<<"\n\n*****Searching Element*****\n";
     cout<<"\nEnter Item value to be search : ";
     cin>>item;
     for(i=0;i<n;i++){
     if(a[i] == item){
        found=i+1;
        break;
     }
     }
     if(found==-1)
     cout<<"\nSearch NOT FOUND\n";
     else
     cout<<"\nSearch is FOUND at "<<found<<" location\n";

}


void Qoperation()
{
    Q q1;
    int n,i,ch,k=0;
    cout<<"Enter number of elements";
    cin>>n;
            cout<<"Enter the Elements";
    for(i=0;i<n;i++)
    {       cin>>k;
            q1.insert(k);
            }
            do{
            cout<<"1.insert\n2.delete\n3.exit\nEnter your choice:";
            cin>>ch;
            switch(ch){
            case 1:{cout<<"Enter an element to insert";
                                cin>>k;
                                q1.insert(k);
                                break;}
            case 2: {int removed = q1.remov();
                        cout<<"Deleted Element after Deletion operation is"<<removed<<"\n";
                        break;}
    case 3:exit(0);
            default: cout<<"Entered the wrong choice";
}}while(1);

}

void stackoperation()
{

      stk stack1;
    int i,choice,e;
  while(1)
  {
    cout<<"Enter 1-push; 2-pop; 3-display\n";
    cin>>choice;
    switch(choice)
    {
        case 1: {cout<<"Enter the element to be pushed\n";
```

```
                cin>>e;
                stack1.push(e);
                break;}
        case 2: {cout<<"The element popped is\n";
                cout<<stack1.pop();
                break;}
        case 3: {cout<<"The elements in the stack are\n";
                 stack1.display();
                 break;}
        default: cout<<"Enter proper choice";
    }

      cout<<"Enter 0 to exit\n";
      cin>>choice;
      if(choice==0)
      exit(0);
  }
}

void linkedlistoperation()
{
    List l1;
    l1.createit();
    l1.insertit();
    l1.insertit();
    l1.insertit();
    l1.displaylist();
    l1.deleteit();
    l1.displaylist();
}

void arrayoperations()
{
   int choice;
   array obj;
   obj.init();
   while(1){

   cout<<"ALL ARRAY OPERATIONS\n\n";
   cout<<"1) Traverse (Display Process)\n";
   cout<<"2) Insert\n";
   cout<<"3) Delete\n";
   cout<<"4) Search\n";
   cout<<"5) Exit\n";
   cout<<"Enter your Choice :  ";
   cin>>choice;
   switch(choice){
     case 1 :  obj.disp();
           break;
     case 2 :  obj.insert();
           break;
     case 3 :  obj.del();
           break;
     case 4 :  obj.search();
           break;
     case 5 :   exit(0);
     default:  cout<<"\n\n\t\tInvalid Choice\n\n";

           break;
   }
 }
}
void setoperation()
{
    setuser s;
    s.display();
    s.search();
    s.eraseele();
}
int main()
{

   int DS;
    cout<<"Enter the datastructure you want to handle\n";
    cout<<"1-Stack;2-Queue;3-Linked List; 4-array; 5-set\n";
    cin>>DS;

    switch(DS)
    {
        case 1: stackoperation();
                break;
        case 2: Qoperation();
                break;
        case 3: linkedlistoperation();
                break;
        case 4: arrayoperations();
                break;
        case 5: setoperation();
                break;
    }
    return 0;
}
```

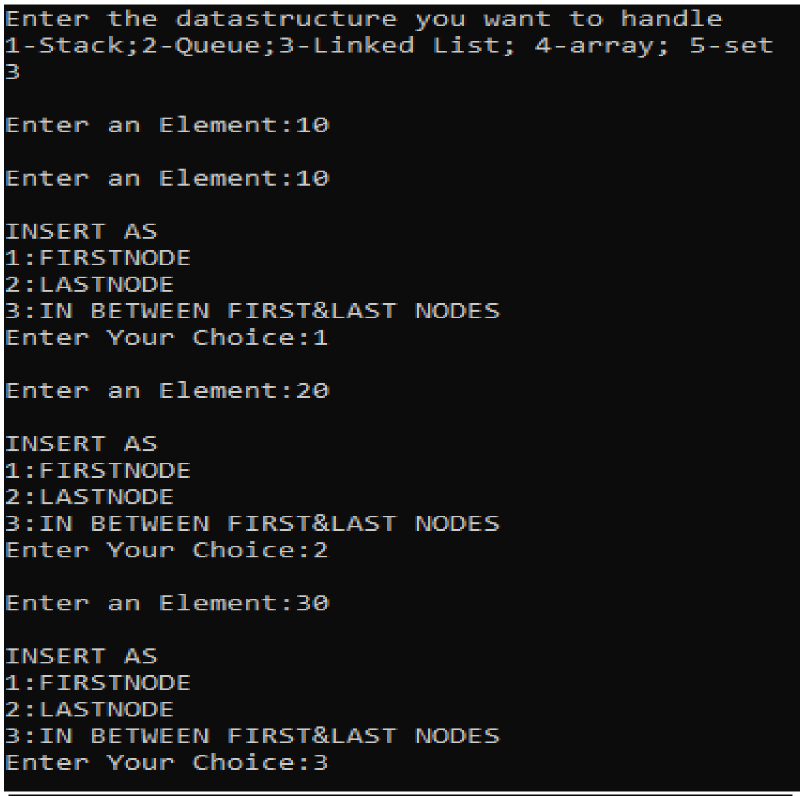
Output:



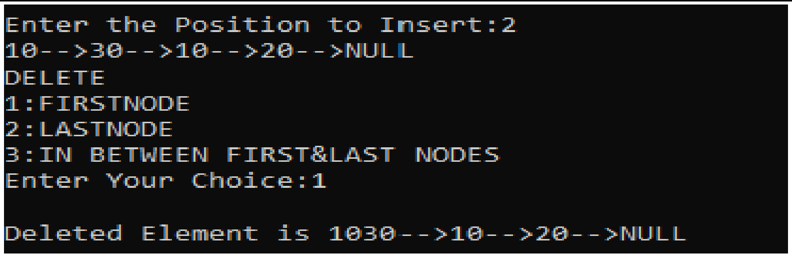
```
1-Stack;2-Queue;3-Linked List; 4-array; 5-set
1
Enter 1-push; 2-pop; 3-display
1
Enter the element to be pushed
20
Enter 0 to exit
2
Enter 1-push; 2-pop; 3-display
1
Enter the element to be pushed
20
Enter 0 to exit
3
Enter 1-push; 2-pop; 3-display
3
The elements in the stack are
20      20      Enter 0 to exit
2
Enter 1-push; 2-pop; 3-display
2
The element popped is
20Enter 0 to exit

3
Enter 1-push; 2-pop; 3-display
3
The elements in the stack are
20      Enter 0 to exit
```



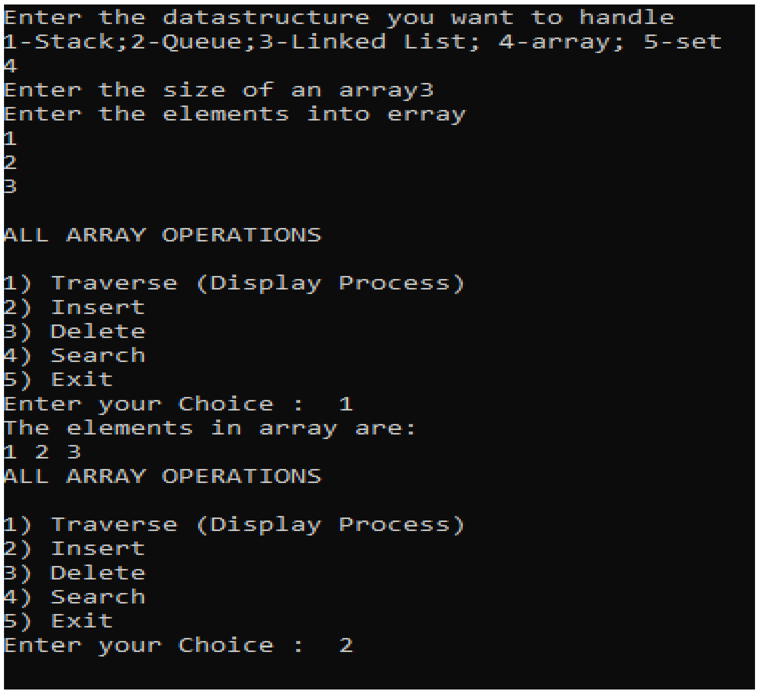
```
Enter the datastructure you want to handle
1-Stack;2-Queue;3-Linked List; 4-array; 5-set
2
Enter number of elements3
Enter the Elements10
Queue elements are : 10
20
Queue elements are : 10 20
30
Queue elements are : 10 20 30
1.insert
2.delete
3.exit
Enter your choice:2
Queue elements are : 20 30
Deleted Element after Deletion operation is10
1.insert
2.delete
3.exit
Enter your choice:3
```



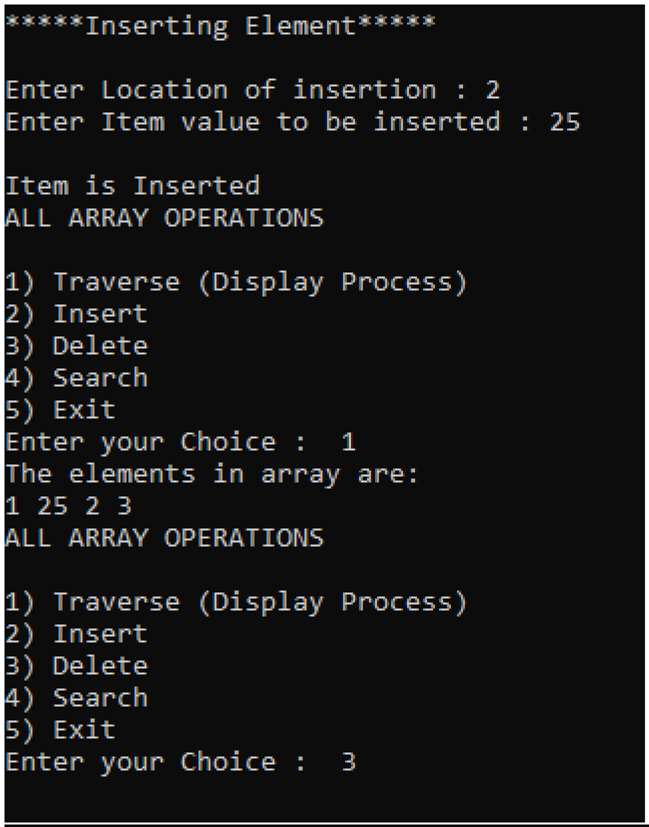
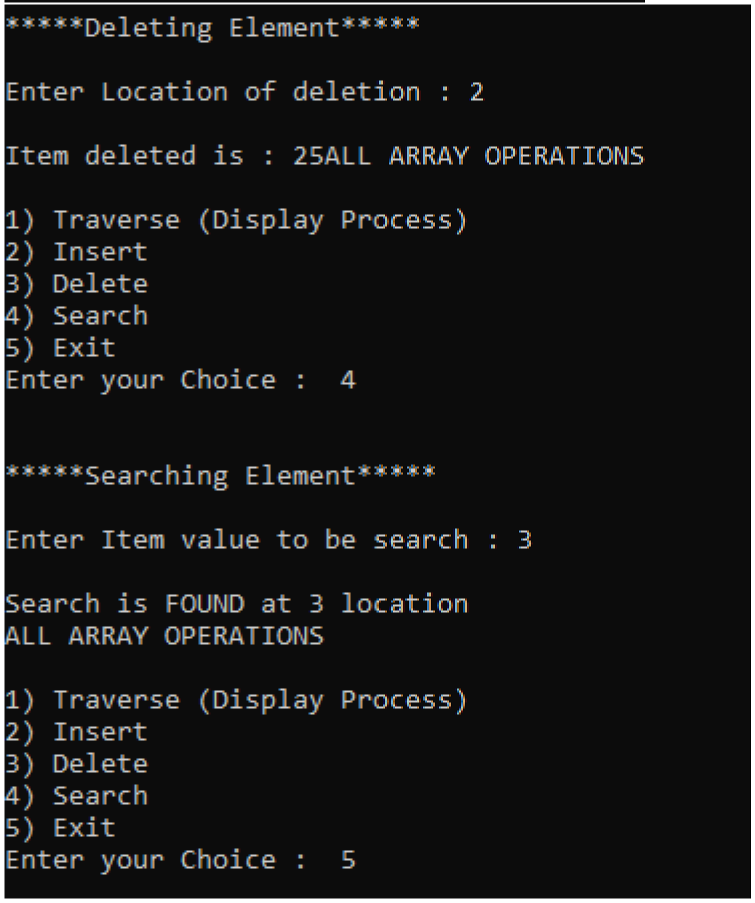
```
Enter the datastructure you want to handle
1-Stack;2-Queue;3-Linked List; 4-array; 5-set
3

Enter an Element:10

Enter an Element:10

INSERT AS
1:FIRSTNODE
2:LASTNODE
3:IN BETWEEN FIRST&LAST NODES
Enter Your Choice:1

Enter an Element:20

INSERT AS
1:FIRSTNODE
2:LASTNODE
3:IN BETWEEN FIRST&LAST NODES
Enter Your Choice:2

Enter an Element:30

INSERT AS
1:FIRSTNODE
2:LASTNODE
3:IN BETWEEN FIRST&LAST NODES
Enter Your Choice:3
```



```
Enter the Position to Insert:2
10-->30-->10-->20-->NULL
DELETE
1:FIRSTNODE
2:LASTNODE
3:IN BETWEEN FIRST&LAST NODES
Enter Your Choice:1

Deleted Element is 1030-->10-->20-->NULL
```



```
Enter the datastructure you want to handle
1-Stack;2-Queue;3-Linked List; 4-array; 5-set
4
Enter the size of an array3
Enter the elements into erray
1
2
3

ALL ARRAY OPERATIONS

1) Traverse (Display Process)
2) Insert
3) Delete
4) Search
5) Exit
Enter your Choice :  1
The elements in array are:
1 2 3
ALL ARRAY OPERATIONS

1) Traverse (Display Process)
2) Insert
3) Delete
4) Search
5) Exit
Enter your Choice :  2
```


```
*****Inserting Element*****

Enter Location of insertion : 2
Enter Item value to be inserted : 25

Item is Inserted
ALL ARRAY OPERATIONS

1) Traverse (Display Process)
2) Insert
3) Delete
4) Search
5) Exit
Enter your Choice :  1
The elements in array are:
1 25 2 3
ALL ARRAY OPERATIONS

1) Traverse (Display Process)
2) Insert
3) Delete
4) Search
5) Exit
Enter your Choice :  3
```

```
*****Deleting Element*****

Enter Location of deletion : 2

Item deleted is : 25ALL ARRAY OPERATIONS

1) Traverse (Display Process)
2) Insert
3) Delete
4) Search
5) Exit
Enter your Choice :  4


*****Searching Element*****

Enter Item value to be search : 3

Search is FOUND at 3 location
ALL ARRAY OPERATIONS

1) Traverse (Display Process)
2) Insert
3) Delete
4) Search
5) Exit
Enter your Choice :  5
```

Task 4

Task 4a): Define class complex with all possible operators: constructor, destructor, copy constructor, assignment operator and operators >, <, >=, <=, ==, ++ (pre and post), +, +=, ( ), with the data members stored as pointer to integers.

Aim: Write a C++ Program for Define class complex with all possible operators: constructor, destructor, copy constructor, assignment operator and operators >, <, >=, <=, ==, ++ (pre and post), +, +=, ( ), with the data members stored as pointer to integers.

Program:

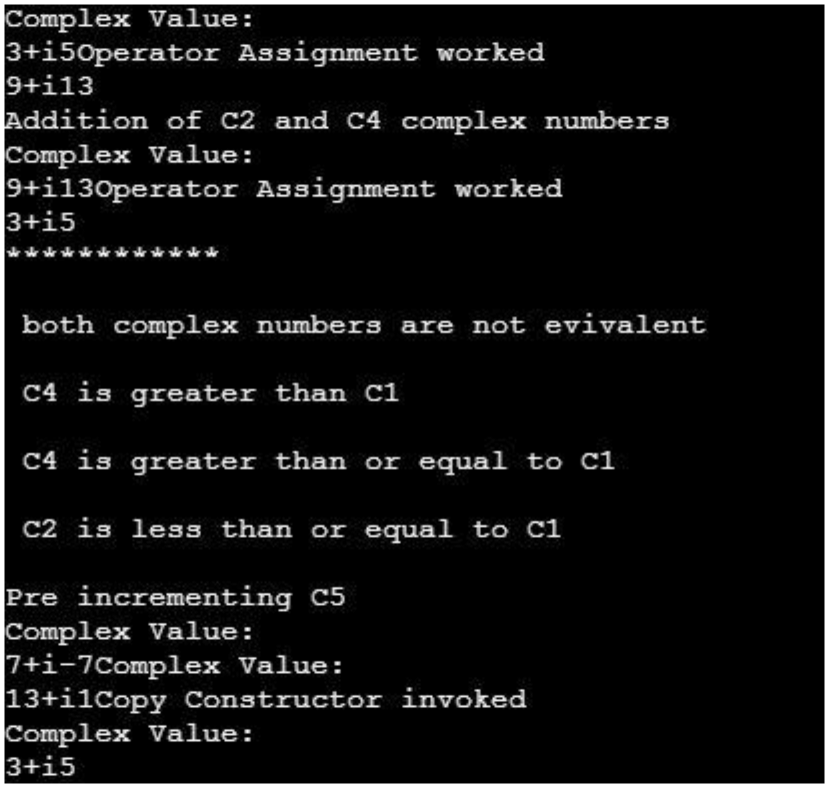
```
#include <iostream>
#include<cmath>
using namespace std;
class Complex{
    private:
      int *real;
      int *imag;
    public:
        Complex()
        {
            real = new int(sizeof(int));
            imag =new int(sizeof(int));
        }
        Complex (int x, int y)
        {
            real = new int(sizeof(int));
            *real=x;
            imag =new int(sizeof(int));
            *imag = y;
        }
        Complex(const Complex &c)  //Copy Constructor
        {
            cout<<"Copy Constructor invoked\n";
            *this->real= *c.real;
           *this->imag=*c.imag;

        }
        ~Complex()
        {
          free(real);
          free(imag);

        }
        void display()
```

```
{
   cout<<"Complex Value:\n"<<*real<<"+i"<<*imag;
}
void read()
{ int i,r;
   cout<< "enter Real Value\n";
   cin>>r;
   cout<<"enter Imaginary Value\n";
   cin>>i;
   *real=r;
   *imag=i;
}
int getreal()
{
   return *real;
}
int getimag()
{
   return *imag;
}
void operator =(const Complex &c) //Assignment Operator
{
   *this->real= *c.real;
   *this->imag= *c.imag;
   cout<<"Operator Assignment worked\n";
   cout<<*this->real<<"+i"<<*this->imag;
}

Complex operator +(const Complex &C)
{
   Complex Ctemp;

   *Ctemp.real= *this->real + *C.real;
   *Ctemp.imag= *this->imag+ *C.imag;
   return Ctemp;
}

Complex operator +=(const Complex &C)
{


   *this->real=*this->real + *C.real;
   *this->imag= *this->imag+ *C.imag;

}
```

```
        void operator ++()
        {

             *this->real=*this->real+1;
             *this->imag=*this->imag+1;

        }

       int operator ==(const Complex &c)const //Comparison Operator
        {  int mod1,mod2;

        cout<<"\n************\n";
        mod1 = sqrt((*this->real)*(*this->real) + (*this->imag)*(*this->imag));
            mod2 = sqrt((*c.real)*(*c.real)+(*c.imag)*(*c.imag));

            if(mod1==mod2)
               return 1;
            else
               return 0;
        }

        friend bool operator>(Complex &a,Complex &b);
        friend bool operator>=(Complex &a,Complex &b);
        friend bool operator<=(Complex &a,Complex &b);
};

bool operator>(Complex &a,Complex &b)
{
    double sum1=0,sum2=0;
    sum1=sqrt(pow(*a.real,2)+pow(*a.imag,2));
    sum2=sqrt(pow(*b.imag,2)+pow(*b.real,2));
    if(sum1>sum2){
        return true;
    }else{
        return false;
        }
}


bool operator>=(Complex &a,Complex &b)
{
    double sum1=0,sum2=0;
    sum1=sqrt(pow(*a.real,2)+pow(*a.imag,2));
    sum2=sqrt(pow(*b.imag,2)+pow(*b.real,2));
    if(sum1<sum2){
        return false;
    }else{
        return true;
        }
}

bool operator<=(Complex &a,Complex &b)
{
    double sum1=0,sum2=0;
    sum1=sqrt(pow(*a.real,2)+pow(*a.imag,2));
    sum2=sqrt(pow(*b.imag,2)+pow(*b.real,2));
    if(sum1<=sum2){
        return true;
    }else{
        return false;
        }
}
int main()
{
    Complex c1;
    Complex c2(3,5);
    Complex c4(6,8),c5(6,-8);
    //c1.read();
    //c1.display();

    c2.display();
    c1=c2+c4;
           cout<<"\nAddition of C2 and C4 complex numbers\n";
           c1.display();
    c1 = c2; //assignment operator
    if(c4==c1)
      cout<<"\n both complex numbers are evivalent\n";
     else
      cout<<"\n both complex numbers are not evivalent\n";


    if(c4>c1)
      cout<<"\n C4 is greater than C1\n";
     else
      cout<<"\n C1 is greater than C4\n";

      if(c4>=c1)
      cout<<"\n C4 is greater than or equal to C1\n";
     else
      cout<<"\n C1 is greater than C4\n";

      if(c2<=c4)
```

```
  cout<<"\n C2 is less than or equal to C1\n";
 else
  cout<<"\n C4 is greater than C2\n";

      cout<<"\nPre incrementing C5\n";
       ++c5;
c5.display();

c5+=c4;
c5.display();
Complex c3=c1;
c3.display();


return 0;

}
```

Output:



```
Complex Value:
3+i5Operator Assignment worked
9+i13
Addition of C2 and C4 complex numbers
Complex Value:
9+i13Operator Assignment worked
3+i5
************

 both complex numbers are not evivalent

 C4 is greater than C1

 C4 is greater than or equal to C1

 C2 is less than or equal to C1

Pre incrementing C5
Complex Value:
7+i-7Complex Value:
13+i1Copy Constructor invoked
Complex Value:
3+i5
```

Task 4b): Define class vector of integers with all possible operations like constructor, destructor, copy constructor and assignment operators>, <, >=, <=, ==, ++ (pre and post), +, +=, ( )

Aim: Write a Program for Define class vector of integers with all possible operations like constructor, destructor, copy constructor and assignment operators>, <, >=, <=, ==, ++ (pre and post), +, +=, ( )

Program:

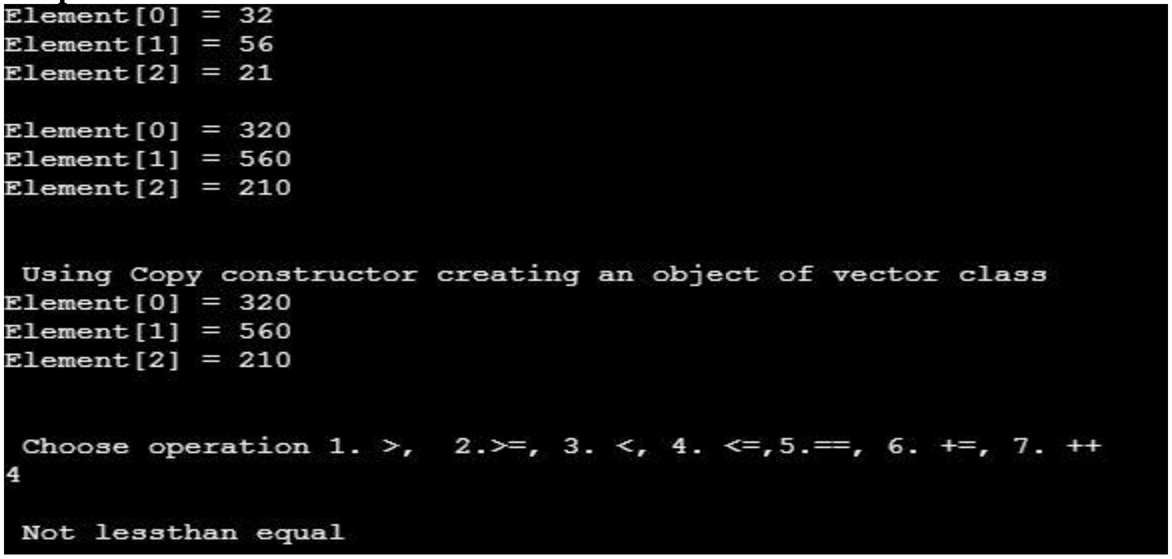
```
#include <iostream>
#include <vector>
using namespace std;

class Object
{
   private:
                  std::vector<int> m_VectorOfInts;
         public:
                  Object()
                  {

                  }
                  ~Object()
                  {

                  }

                  Object(const Object  &O1)
                  {

                           m_VectorOfInts=O1.m_VectorOfInts;

                  }


                  int operator>(Object const &O1)
                  {
                     for( unsigned int i = 0; i < O1.m_VectorOfInts.size(); i++ )
                       {
                          if(m_VectorOfInts[i]<=O1.m_VectorOfInts[i])
                            {
                              return 0;
                            }
                       }
   cout<<"\n The vector is greater\n";
   return 1;
}

int operator>=(Object const &O1)
{
   for( unsigned int i = 0; i < O1.m_VectorOfInts.size(); i++ )
     {
        if(m_VectorOfInts[i]<O1.m_VectorOfInts[i])
          {
           return 0;
          }
     }
   cout<<"\n The vector is greater\n";
   return 1;
}

int operator<(Object const &O1)
{
   for( unsigned int i = 0; i < O1.m_VectorOfInts.size(); i++ )
     {
        if(m_VectorOfInts[i]>=O1.m_VectorOfInts[i])
          {
           return 0;
          }
     }
   cout<<"\n The vector is greater\n";
   return 1;
}

int operator<=(Object const &O1)
{
   for( unsigned int i = 0; i < O1.m_VectorOfInts.size(); i++ )
     {
        if(m_VectorOfInts[i]>O1.m_VectorOfInts[i])
          {
           return 0;
          }
     }
   cout<<"\n The vector is greater\n";
   return 1;
}

int operator==(Object const &O1)
{
   for( unsigned int i = 0; i < O1.m_VectorOfInts.size(); i++ )
```

```
        {
            if(m_VectorOfInts[i]!=O1.m_VectorOfInts[i])
              {
                return 0;
              }
        }
    cout<<"\n The vector is greater\n";
    return 1;
}

void operator +=(Object const &O1)
{
    for( unsigned int i = 0; i < O1.m_VectorOfInts.size(); i++ )
        {
            m_VectorOfInts[i]= m_VectorOfInts[i] + O1.m_VectorOfInts[i];
        }

}



            void operator ++()
{
    for( unsigned int i = 0; i <m_VectorOfInts.size(); i++ )
        {
            m_VectorOfInts[i]= m_VectorOfInts[i] +1;
        }

}

void AddInt(int num)
{
            m_VectorOfInts.push_back(num);
}
std::vector<int> GetCopyOfVector()
{
            return m_VectorOfInts;
}
void DisplayVectorContents()
{
            for( unsigned int i = 0; i < m_VectorOfInts.size(); i++ )
            {
                        std::cout << "Element[" << i << "] = " << m_VectorOfInts[i] <<
std::endl;
            }
            std::cout << std::endl;
}
```

```
};

int main()
{
    int choice;
            // Create our class an add a few ints
            Object obj,obj1;
            obj.AddInt(32);
            obj.AddInt(56);
            obj.AddInt(21);
    obj1.AddInt(320);
            obj1.AddInt(560);
            obj1.AddInt(210);
            // Display the vector contents so far
            obj.DisplayVectorContents();
    obj1.DisplayVectorContents();
    cout<<"\n Using Copy constructor creating an object of vector class\n";
Object objCopyCon(obj1);
 objCopyCon.DisplayVectorContents();
         cout<<"\n Choose operation 1. >,  2.>=, 3. <, 4. <=,5.==, 6. +=, 7. ++ \n";
          cin>>choice;
          switch (choice)
          {
              case 1:   if(obj1>obj)
                        {
                            cout<<"\n Yes greater\n";
                        }
                        else
                            cout<<"\n Not greater than\n";
                        break;
      case 2:      if(obj1>=obj)
                        {
                            cout<<"\n Yes greater\n";
                        }
                        else
                            cout<<"\n Not greater than\n";
                        break;

              case 3:   if(obj1<obj)
                        {
                            cout<<"\n Yes Smaller\n";
                        }
                        else
                            cout<<"\n Not smaller than\n";
```

```
                break;

           case 4:   if(obj1<=obj)
                {
                   cout<<"\n Yes lessthan equal\n";
                }
                else
                   cout<<"\n Not lessthan equal\n";
                break;

           case 5:   if(obj1==obj)
                {
                   cout<<"\n Yes Equal\n";
                }
                else
                   cout<<"\n Not Equal\n";
                break;

           case 6:  obj1+=obj;
               obj1.DisplayVectorContents();
              break;
           case 7:  ++obj;
               cout<<"\n values after pre incrementation of obj\n";
               obj.DisplayVectorContents();
              break;

           default: cout<<"\n enter proper choice\n";
         }
          return 0;
    }
```

Output:



```
Element[0] = 32
Element[1] = 56
Element[2] = 21

Element[0] = 320
Element[1] = 560
Element[2] = 210


 Using Copy constructor creating an object of vector class
Element[0] = 320
Element[1] = 560
Element[2] = 210


 Choose operation 1. >,  2.>=, 3. <, 4. <=,5.==, 6. +=, 7. ++
4

 Not lessthan equal
```

Task 5:

Task 5a): Define class matrix of integers with all possible operations like constructor, destructor, copy constructor and assignment operators>, <, >=, <=, ==, ++ (pre and post), +, +=, ( ).

Aim: Write a Program for Define class matrix of integers with all possible operations like constructor, destructor, copy constructor and assignment operators>, <, >=, <=, ==, ++ (pre and post), +, +=, ( ).

Program:

```
#include<iostream>
using namespace std;
#include<cstdlib>
#include<new>
class matrix
{

  int **array;
  int row;
  int column;

public:

  //default constructor
    matrix ()
  {

    array = NULL;
    row = 0;
    column = 0;
  }

  //single parameter constructor(square matrix)
  matrix (int length)
  {

    int i, j;
    array = (int **) new int[length];
    row = length;
    column = length;
    for (i = 0; i < length; i++)
      array[i] = new int[length];

    for (i = 0; i < length; i++)
    for (j = 0; j < length; j++)
    array[i][j] = 0;
}


//double parameterized constructor(rectangular matrix)
matrix (int r, int c)
{
  int i, j;
  row = r;
  column = c;
  array = (int **) new int[r];

  for (i = 0; i < r; i++)
    array[i] = new int[c];

  for (i = 0; i < r; i++)
    for (j = 0; j < c; j++)
    array[i][j] = 0;
}

//copy constructor

matrix (const matrix & c)
{
  int i, j;
  array = (int **) new int[c.row];

  for (i = 0; i < c.row; i++)
    array[i] = new int[c.column];
  row = c.row;
  column = c.column;
  for (i = 0; i < c.row; i++)
    for (j = 0; j < c.column; j++)
    array[i][j] = c.array[i][j];
}

//destructor
~matrix ()
{
  delete[]array;
}

//OVerloading matrix "+" operation

matrix operator+ (const matrix & c)
```

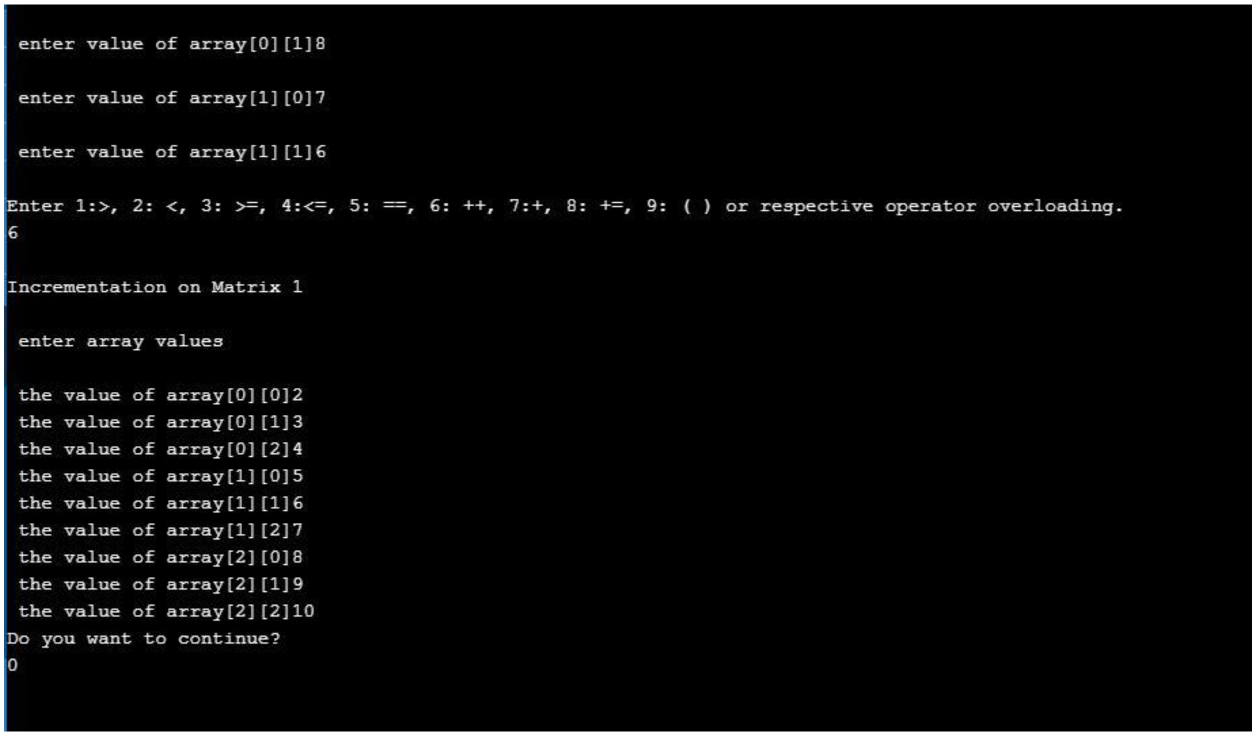
```
{
  int i, j;
  matrix temp;
  if (row != c.row || column != c.column)
    {
    cout << "\n addition is not possible";
    exit (1);
    }
  temp.row = row;
  temp.column = column;
  temp.array = (int **) new int[row];
  for (i = 0; i < row; i++)
    temp.array[i] = new int[column];
  for (i = 0; i < row; i++)
    for (j = 0; j < column; j++)
    temp.array[i][j] = array[i][j] + c.array[i][j];

  return temp;
}

void operator+= (const matrix & c)
{
  int i, j;
  matrix temp;
  if (row != c.row || column != c.column)
    {
    cout << "\n addition is not possible";
    exit (1);
    }

  for (i = 0; i < row; i++)
    for (j = 0; j < column; j++)
    array[i][j] = array[i][j] + c.array[i][j];

}

matrix operator= (const matrix & mat)
{
  int i, j;

  if (row != mat.row || column != mat.column)
    {
    cout << "\n\nAssignment of different matrices is not possible..\n";
    exit (1);
    }
  for (i = 0; i < row; i++)
    for (j = 0; j < column; j++)
    array[i][j] = mat.array[i][j];

  return (*this);
}

int operator == (const matrix & mat)
{
  int i, j, E = 1;

  if (row != mat.row || column != mat.column)
    {
    cout << "\n\ndifferent matrices..\n";
    exit (1);
    }
  else
    {
    for (i = 0; i < row; i++)
      for (j = 0; j < column; j++)
        if (array[i][j] != mat.array[i][j])
          E = 0;
    }
  return E;
}

int operator >= (const matrix & mat)
{
  int i, j, GE = 1;

  if (row != mat.row || column != mat.column)
    {
    cout << "\n\ndifferent matrices..\n";
    exit (1);
    }
  else
    {
    for (i = 0; i < row; i++)
      for (j = 0; j < column; j++)
        if (array[i][j] < mat.array[i][j])
          GE = 0;
    }
  return GE;
}

int operator<= (const matrix & mat)
```

```
{
  int i, j, LE = 1;

  if (row != mat.row || column != mat.column)
    {
    cout << "\n\ndifferent matrices..\n";
    exit (1);
    }
  else
    {
    for (i = 0; i < row; i++)
      for (j = 0; j < column; j++)
        if (array[i][j] > mat.array[i][j])
          LE = 0;
    }
  return LE;
}

int operator> (const matrix & mat)
{
  int i, j, G = 1;

  if (row != mat.row || column != mat.column)
    {
    cout << "\n\ndifferent matrices..\n";
    exit (1);
    }
  else
    {
    for (i = 0; i < row; i++)
      for (j = 0; j < column; j++)
        if (array[i][j] <= mat.array[i][j])
          G = 0;
    }
  return G;
}

int operator< (const matrix & mat)
{
  int i, j, L = 1;

  if (row != mat.row || column != mat.column)
    {
    cout << "\n\ndifferent matrices..\n";
    exit (1);
    }
  else
    {
    for (i = 0; i < row; i++)
      for (j = 0; j < column; j++)
        if (array[i][j] > mat.array[i][j])
          L = 0;
    }
  return L;
}
matrix operator- (const matrix & c)
{
  int i, j;
  matrix temp;
  if (row != c.row || column != c.column)
    {
    cout << "\n addition is not possible";
    exit (1);
    }
  temp.array = (int **) new int[row];
  for (i = 0; i < row; i++)
    temp.array[i] = new int[column];
  for (i = 0; i < row; i++)
    for (j = 0; j < column; j++)
    temp.array[i][j] = array[i][j] - c.array[i][j];
  temp.row = row;
  temp.column = column;
  return temp;
}
matrix operator+ (int value)
{
  int i, j;
  matrix temp;
  temp.array = (int **) new int[row];
  for (i = 0; i < row; i++)
    temp.array[i] = new int[column];
  for (i = 0; i < row; i++)
    for (j = 0; j < column; j++)
    temp.array[i][j] = array[i][j] + value;
  temp.row = row;
  temp.column = column;
  return temp;
}

void operator ++ ()
{
  int i = 0, j = 0;
  for (i = 0; i < row; i++)
    for (j = 0; j < column; j++)
    ++(array[i][j]);

}

matrix operator- (int value)
{
  int i, j;
  matrix temp;
  temp.array = (int **) new int[row];
  for (i = 0; i < row; i++)
    temp.array[i] = new int[column];
  for (i = 0; i < row; i++)
    for (j = 0; j < column; j++)
    temp.array[i][j] = array[i][j] - value;
  temp.row = row;
  temp.column = column;
  return temp;
}


void *operator   new[] (size_t size)
{
  void *pointer = malloc (size);
  return pointer;
}

void operator   delete[] (void *pointer)
{
  free (pointer);
}


matrix operator  () (int r, int c, int ut)
{
  int i, j;
  matrix temp;
  temp.row = r;
  temp.column = c;
  temp.array = (int **) new int[r];

  for (i = 0; i < r; i++)
    temp.array[i] = new int[c];

  if (ut = 1)
```

```
      {
      for (i = 0; i < r; i++)
        for (j = 0; j < c; j++)
          if (i < j)
            temp.array[i][j] = 0;
          else
            temp.array[i][j] = 1;
      }

    return temp;
  }
  friend matrix operator+ (int value, matrix c);

  friend matrix operator- (int value, matrix c);

  friend istream & operator>> (istream & cin, matrix & c);

  friend ostream & operator<< (ostream & cout, matrix & c);


};

istream & operator>> (istream & cin, matrix & c)
{

  int i, j;
  cout << "\n enter array values" << endl;
  for (i = 0; i < c.row; i++)
    {
      for (j = 0; j < c.column; j++)
      {

        cout << "\n enter value of array[" << i << "][" << j << "]";
        cin >> c.array[i][j];
      }
    }
  return cin;
}

ostream & operator<< (ostream & cout, matrix & c)
{

  int i, j;
  cout << "\n enter array values" << endl;
  for (i = 0; i < c.row; i++)
    {
      for (j = 0; j < c.column; j++)
      {

        cout << "\n the value of array[" << i << "][" << j << "]";
        cout << c.array[i][j];
      }
    }
  return cout;
}

matrix
operator+ (int value, matrix c)
{

  matrix temp;
  int i, j;
  temp.array = (int **) new int[c.row];
  for (i = 0; i < c.row; i++)
    temp.array[i] = new int[c.column];
  temp.row = c.row;
  temp.column = c.column;

  for (i = 0; i < c.row; i++)
    for (j = 0; j < c.column; j++)
      temp.array[i][j] = value + c.array[i][j];
  return temp;
}
matrixoperator- (int value, matrix c)
{
  matrix temp;
  int i, j;
  temp.array = (int **) new int[c.row];
  for (i = 0; i < c.row; i++)
    temp.array[i] = new int[c.column];
  temp.row = c.row;
  temp.column = c.column;
for (i = 0; i < c.row; i++)
    for (j = 0; j < c.column; j++)
      temp.array[i][j] = value - c.array[i][j];
  return temp;
}

int main ()
{
   int choice,Y=1;
   matrix m1 (3), m2 (2), m3 (2), m4(2),m(3),m5(3),m6(3);
```

```
 cout<<"\n enter matrix m1 values";
 cin>>m1;
 cout<<"\nEnter second matrix\n";
 cin>>m2;
while(Y)
 {
cout<< "\nEnter 1:>, 2: <, 3: >=, 4:<=, 5: ==, 6: ++, 7:+, 8: +=, 9: ( ) or respective operator
overloading.\n";
 cin>>choice;
 switch(choice)
 {
   case 1:
           if(m1>m2)
             cout<<"M1 is greater\n";
           else
             cout<<"M2 is greater\n";
           break;
   case 2:
           if(m1<m2)
             cout<<"M1 is smaller\n";
           else
             cout<<"M2 is smaller\n";
           break;

   case 3:  m3 = m1 + m2;
            if(m3>=m1)
             cout<<"M3 found to be >= than M1\n";
            else
             cout<<"M3 found to be not >= than M1\n";
            break;
   case 4:
           m3 = m1 + m2;
            if(m3<=m1)
             cout<<"M3 found to be <= than M1\n";
            else
             cout<<"M3 found to be not <= than M1\n";
                break;
   case 5:
           if(m3==m1)
             cout<<"M3 and M1 are equal\n";
           else
           cout<<"M3 and M1 are not equal\n";

           break;
   case 6:
          cout<<"\nIncrementation on Matrix 1\n";
```

```
                ++m1;
                cout<<m1;
                break;
         case 7:  cout<<"OVerloading + operator on matrces\n";
                  m4 = m1 + m2;
                  cout<<"Matrix 4 created by adding M1 and M2 using + is:\n";
                break;
         case 8: cout<<"overloading shorthand += symbol\n";
                 m5 += m1;
                 cout<<"Matrix 5 created and using += ith M1 gives:\n";
                 cout<<m5;
                 break;
         case 9: cout<<"Overloading () operator to create upper triangle Matrix 6\n";
                 m6 = m(3,3,1);
                 cout<<m6;
                 break;
      }
       cout<<"\nDo you want to continue?\n";
       cin>>Y;

         if(Y==0)
         break;
      }
      return 0;
      }
```
Output:



```
 enter value of array[0][1]8

 enter value of array[1][0]7

 enter value of array[1][1]6

Enter 1:>, 2: <, 3: >=, 4:<=, 5: ==, 6: ++, 7:+, 8: +=, 9: ( ) or respective operator overloading.
6

Incrementation on Matrix 1

 enter array values

 the value of array[0][0]2
 the value of array[0][1]3
 the value of array[0][2]4
 the value of array[1][0]5
 the value of array[1][1]6
 the value of array[1][2]7
 the value of array[2][0]8
 the value of array[2][1]9
 the value of array[2][2]10
Do you want to continue?
0
```

Task 5b): Define class matrix of integers using vector, with all possible operations like constructor, destructor, copy constructor and assignment operators>, <, >=, <=, ==, ++ (pre and post), +, +=, ( ).

Aim: Write a C++ Program for Define class matrix of integers using vector, with all possible operations like constructor, destructor, copy constructor and assignment operators>, <, >=, <=, ==, ++ (pre and post), +, +=, ( ).

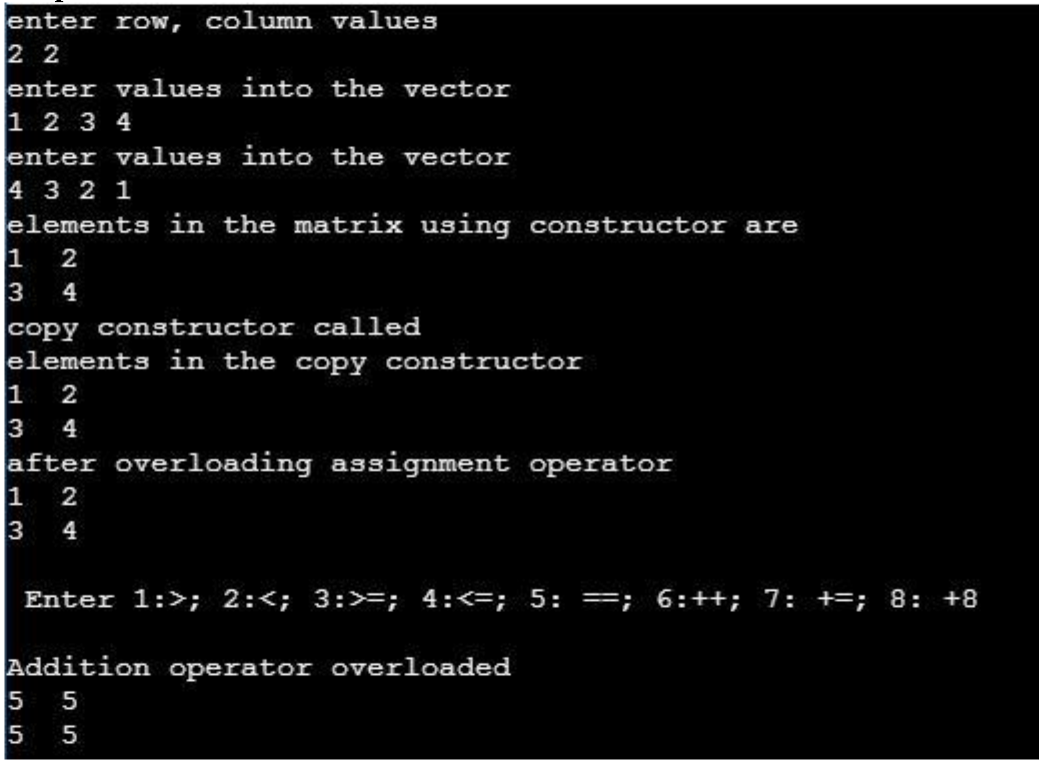
Program:

```
#include<iostream>
#include<vector>
using namespace std;
class Matrix
{
  int x,y;                // redundant class data erased
  vector< vector <int> > v;
  public:
  Matrix()
  {
     x=0; y=0;
     v = vector< vector<int> >(x,vector <int>(y, 0));
     for(int i = 0; i < x; ++i)
     {
        for (int j = 0; j < y; ++j)
        {
           v[i][j]=0;
        }
     }
  }
  Matrix(int m, int n) // Main constructor
  {
     x=m;
     y=n;
     v = vector< vector<int> >(x,vector <int>(y, 0));
     cout<< "enter values into the vector" << endl;
     for(int i = 0; i < x; ++i)
     {
        for (int j = 0; j < y; ++j)
        {
           cin >> v[i][j];
        }
     }
  }
```

```
Matrix(const Matrix &t)
{
    x=t.x;
    y=t.y;
    v = vector< vector<int> >(x,vector <int>(y, 0));
    //v=t.v;
     for (int i = 0; i < x; ++i)
    {
        for (int j = 0; j < y; ++j)
        {
            v[i][j] = t.v[i][j];
        }
    }
    cout << "copy constructor called" << endl;
}
~Matrix()
{
    //cout << "destructor called" << endl;
}
bool operator >(const Matrix r)
{
    if(x!=r.x||y!=r.y)
      {
          cout<<"\nSize Mismatch\n";
          return false;
      }
    else
    {
        for (int i = 0; i < x; ++i)
    {
        for (int j = 0; j < y; ++j)
        {
            if(v[i][j] <= r.v[i][j])
                return false;
        }       }       }

    return true;
}

bool operator <(const Matrix r)
{
    if(x!=r.x||y!=r.y)
      {
          cout<<"\nSize Mismatch\n";
          return false;
      }
    else
    {
        for (int i = 0; i < x; ++i)
    {
        for (int j = 0; j < y; ++j)
        {
            if(v[i][j] >= r.v[i][j])
                return false;
        }
    }

    }

    return true;
}


bool operator >=(const Matrix r)
{
    if(x!=r.x||y!=r.y)
      {
          cout<<"\nSize Mismatch\n";
          return false;
      }
    else
    {
        for (int i = 0; i < x; ++i)
    {
        for (int j = 0; j < y; ++j)
        {
            if(v[i][j] < r.v[i][j])
                return false;
        }
    }

    }

    return true;
}

bool operator <=(const Matrix r)
{
    if(x!=r.x||y!=r.y)
      {
          cout<<"\nSize Mismatch\n";
          return false;
      }
    else
    {
        for (int i = 0; i < x; ++i)
    {
        for (int j = 0; j < y; ++j)
        {
            if(v[i][j] > r.v[i][j])
                return false;
        }
    }

    }

    return true;
}


bool operator ==(const Matrix r)
{
    if(x!=r.x||y!=r.y)
      {
          cout<<"\nSize Mismatch\n";
          return false;
      }
    else
    {
        for (int i = 0; i < x; ++i)
    {
        for (int j = 0; j < y; ++j)
        {
            if(v[i][j] != r.v[i][j])
                return false;
        }
    }

    }

    return true;
}


void operator ++()
{

        for (int i = 0; i < x; ++i)
    {
        for (int j = 0; j < y; ++j)
        {
            v[i][j]++;
        }
    }
}

void operator +=(const Matrix r)
{
    if(x!=r.x||y!=r.y)
      {
          cout<<"\nSize Mismatch\n";

      }
    else
    {
        for (int i = 0; i < x; ++i)
    {
        for (int j = 0; j < y; ++j)
        {
            v[i][j] = v[i][j]+ r.v[i][j];

        }
    }
    }
}

Matrix& operator+(const Matrix &r)
{
    if(x!=r.x||y!=r.y)
    {
        cout<<"order mismatch to add\n";
        return *this;
    }

    //Matrix b;
    for (int i = 0; i < x; ++i)
    {
        for (int j = 0; j < y; ++j)
        {
            v[i][j] = v[i][j] + r.v[i][j];
        }
    }
         return *this;
}
```

```
  Matrix& operator=(const Matrix &r)   // needs to output a reference
  {
     if (this == &r)   // check self-assignment
        return *this;

     x = r.x;
     y = r.y;

     v = vector< vector<int> >(x,vector <int>(y, 0));
     //vector< vector<int> > b(x,vector <int>(y, 0));
     for (int i = 0; i < x; ++i)
     {
        for (int j = 0; j < y; ++j)
        {
           v[i][j] = r.v[i][j];
        }
     }



     return *this;
  }

  void print()
  {
     for (int i = 0; i < x; i++)
     {
        for (int j = 0; j < y; j++)
        {
           cout << v[i][j] << "  ";
        }
        cout << endl;
     }
  }
};

int main()
{
  int d1,d2,choice;
  cout << "enter row, column values" << endl;
  cin >> d1 >> d2;
  Matrix ob1;
Matrix ob2(d1,d2);
Matrix mat(d1,d2);
cout << "elements in the matrix using constructor are" << endl;
ob2.print();

Matrix ob3(ob2);
cout << "elements in the copy constructor" << endl;
ob3.print();
cout << "after overloading assignment operator" << endl;
Matrix ob4;
ob4=ob2;
ob4.print();


cout<<"\n Enter 1:>; 2:<; 3:>=; 4:<=; 5: ==; 6:++; 7: +=; 8: +";
cin>> choice;
switch(choice)
{
    case 1:  if(mat>ob2)
                cout<<"\nMat is greater\n";
            else
                cout<<"\nMat is smaller\n";
            break;
     case 2:
            if(mat<ob2)
                cout<<"\nMat is smaller\n";
            else
                cout<<"\nMat is greater\n";
            break;

     case 3:
            if(mat>=ob2)
                cout<<"\nMat is greater or equal\n";
            else
                cout<<"\nMat is smaller\n";

             break;
     case 4:
            if(mat<=ob2)
                cout<<"\nMat is smaller or equal\n";
            else
                cout<<"\nMat is greater\n";
             break;
     case 5:
            if(mat==ob2)
                cout<<"\nMat is equal to ob2\n";
                else
                    cout<<"\nMat and ob2 are not equal\n";
                break;
            case 6:   cout<<"\n pre incrementor overloading on mat object\n";
                    ++mat;
                    mat.print();
                break;
            case 7:
                 mat+=ob2;
                 cout<<"\nAdding ob2 to mat using += operator overloading\n";
                 mat.print();
                break;
            case 8:
                cout<<"\nAddition operator overloaded\n";
                ob1 = ob2+mat;
                ob1.print();
                break;
            default: cout<<"\nEnter proper option\n";

        }

        return 0;
    }
```

Output:



```
enter row, column values
2 2
enter values into the vector
1 2 3 4
enter values into the vector
4 3 2 1
elements in the matrix using constructor are
1  2
3  4
copy constructor called
elements in the copy constructor
1  2
3  4
after overloading assignment operator
1  2
3  4

 Enter 1:>; 2:<; 3:>=; 4:<=; 5: ==; 6:++; 7: +=; 8: +8

Addition operator overloaded
5  5
5  5
```

Task-6

Task-6: Define stack and queue inherited from array class, with standard functions and operators

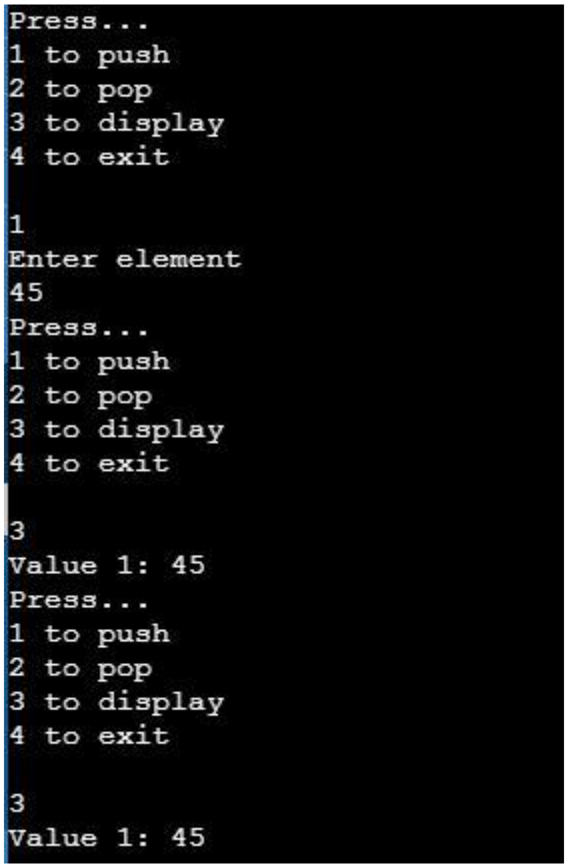
Aim: Write a C++ Program for Define stack and queue inherited from array class, with standard functions and operators

Program:

```
//Stack
#include<iostream>
using namespace std;
class stackdata{
        protected:
                int stackarray[4],top;
                stackdata()
                {
                        top=-1;
                }
                ~stackdata()
                {
                        "Stack destructed!\n";
                }
};

class stackfunction:public stackdata{
        public:
                void push(int n)
                {
                        if (top==3)
                        cout<<"Stack overflow\n";
                        else
                        {
                                top++;
                                stackarray[top]=n;
                        }
                }
                void pop()
                {
                        if (top==-1)
                                cout<<"Stack underflow\n";
                        else
                                {
                                        cout<<stackarray[top]<<" popped successfully\n";
                                        top--;
                }
        }
        void display()
        {
            if(top==-1)
            cout<<"Stack empty\n";
            else
            {
                for (int i=0;i<=top;i++)
                cout<<"Value "<<i+1<<": "<<stackarray[i]<<endl;
            }
        }

};

int main()
{
    stackfunction s1;
    char ch;
    do{


    cout<<"Press...\n"
        <<"1 to push\n"
        <<"2 to pop\n"
        <<"3 to display\n"
        <<"4 to exit\n\n";
    cin>>ch;
    switch(ch)
    {
        case '1':
            int n;
            cout<<"Enter element\n";
            cin>>n;
            s1.push(n);
            break;
        case '2':
        s1.pop();
        break;
        case '3':
        s1.display();
        break;
    }
}
while(ch!='4');
    return 0;
```

```
}
```

Output:



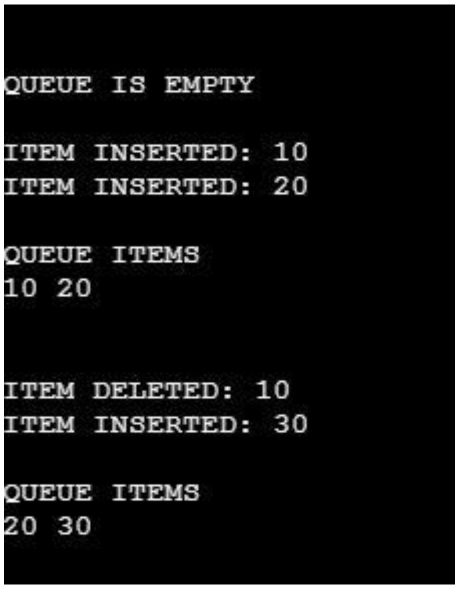
```
Press...
1 to push
2 to pop
3 to display
4 to exit

1
Enter element
45
Press...
1 to push
2 to pop
3 to display
4 to exit

3
Value 1: 45
Press...
1 to push
2 to pop
3 to display
4 to exit

3
Value 1: 45
```

```
//Queue
#include <iostream>
#define MAX 10

using namespace std;

class Queue
{
    int front,rear;
    int queue[MAX];

    public:

    Queue()
    {
        front=rear=-1;
```

```
}

 void qinsert(int item)
 {
        if(rear==MAX-1)
       {
                cout<<"\nQUEUE OVERFLOW";
       }
       else if(front==-1 && rear==-1)
       {
                front=rear=0;
                queue[rear]=item;
                cout<<"\nITEM INSERTED: "<<item;
       }
       else
       {
                rear++;
                queue[rear]=item;
                cout<<"\nITEM INSERTED: "<<item;
       }
 }

 void qdelete()
 {
        int item;

        if(rear==-1)
       {
                 cout<<"\nQUEUE UNDERFLOW";
       }
       else if(front==0 && rear==0)
       {
                 item=queue[front];
                 front=rear=-1;
                 cout<<"\n\nITEM DELETED: "<<item;
       }
       else
       {
                item=queue[front];
                front++;
                cout<<"\n\nITEM DELETED: "<<item;
       }
 }

 void qtraverse()
 {
```

```
                if(front==-1)
                {
                        cout<<"\n\nQUEUE IS EMPTY\n";
                }
                else
                {
                        cout<<"\n\nQUEUE ITEMS\n";
                        for(int i=front;i<=rear;i++)
                        {
                                cout<<queue[i]<<" ";
                        }
                        cout<<endl;
                }
          }
  };

  int main()
  {
   Queue q;
        q.qtraverse();
        q.qinsert(10);
        q.qinsert(20);
        q.qtraverse();
        q.qdelete();
        q.qinsert(30);

        q.qtraverse();
        return 0;
  }
```

Output:



```


QUEUE IS EMPTY

ITEM INSERTED: 10
ITEM INSERTED: 20

QUEUE ITEMS
10 20


ITEM DELETED: 10
ITEM INSERTED: 30

QUEUE ITEMS
20 30
```

Task -7

Task 7a): Define a class called 'array' with data type passed as template type with constructor, destructor, copy constructor and assignment operators and index operator.

Aim: Write a C++ Program for Define a class called 'array' with data type passed as template type with constructor, destructor, copy constructor and assignment operators and index operator.

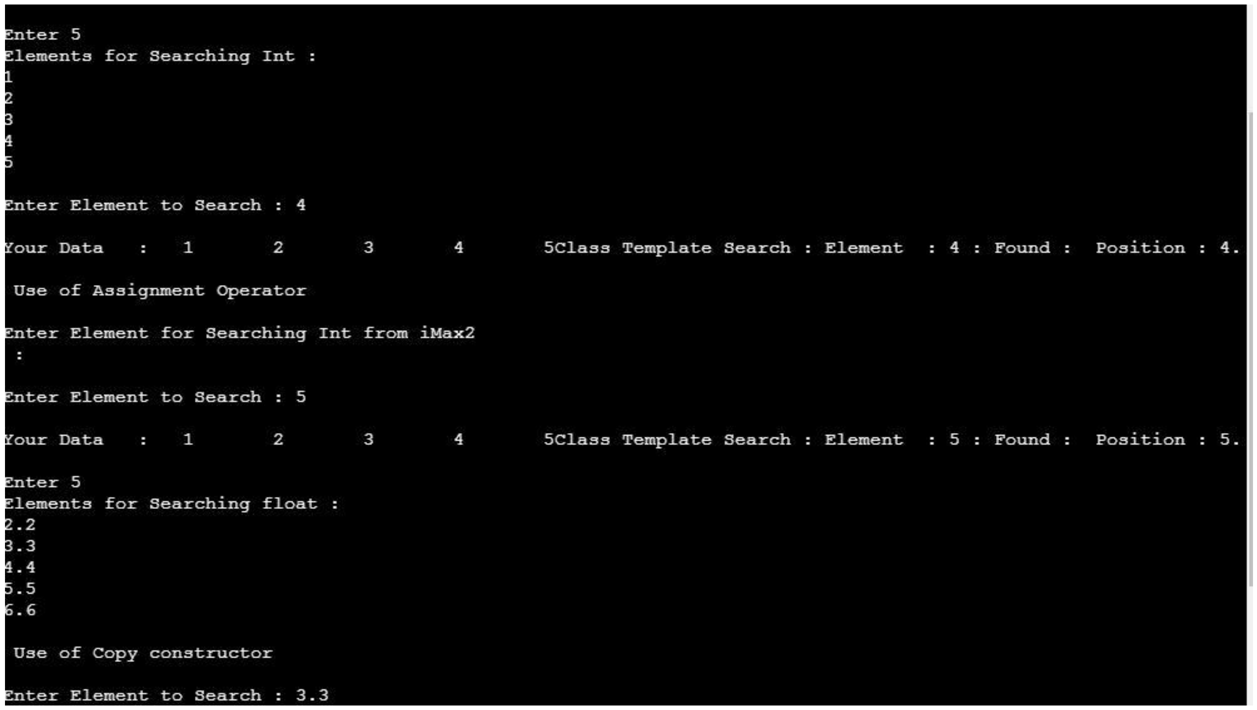
Program:

```
#include <iostream>
#include<conio.h>
#include<stdlib.h>

#define MAX_SIZE 5

using namespace std;

// Template Declaration
template<class T>

// Generic Template Class for Search
class TClassSearch {

        T x[MAX_SIZE];
        T element;

        public:

        TClassSearch()
        {

        }

        TClassSearch(const TClassSearch &D)
        {
           int i = 0;
                 for (i = 0; i < MAX_SIZE; i++)
                    this->x[i]=D.x[i];
        }

        void operator = (TClassSearch D)
        {
           int i = 0;
                 for (i = 0; i < MAX_SIZE; i++)
```

```
                         this->x[i]=D.x[i];

             }

  TClassSearch &operator[] (const int i)
             {
                if(i>=MAX_SIZE)
                {
                cout<<"\narray out of bounds\n";

                    exit(0);
                }
                  return x[i];
             }
    void readElements()
      {
           int i = 0;
                       for (i = 0; i < MAX_SIZE; i++)
        cin >> x[i];
             //x*=t_x; element=t_element;
      }
      void SearchEle()
      {
                       cout << "\nEnter Element to Search : ";
           cin>>element;
              }
T getSearch() {
           int i;
cout << "\nYour Data   :";
           for (i = 0; i < MAX_SIZE; i++) {
        cout << "\t" << x[i];
    }
   /* for : Check elements one by one - Linear */
    for (i = 0; i < MAX_SIZE; i++) {
        /* If for Check element found or not */
        if (x[i] == element) {
            cout << "\Class Template Search : Element  : " << element << " : Found :  Position : " << i
+ 1 << ".\n";
            break;
        }
    }

    if (i == MAX_SIZE)
        cout << "\nSearch Element : " << element << "  : Not Found \n";
            }
};
```

```
int main() {
        int i;
            TClassSearch <int> iMax = TClassSearch<int>();
    TClassSearch <float> fMax = TClassSearch<float>();
        TClassSearch <int> iMax2 = TClassSearch<int>();
        cout << "\nSimple Class Template Array Program Example : Search Number\n";
    cout << "\nEnter " << MAX_SIZE << " \nElements for Searching Int : " << endl;
        iMax.readElements();
    iMax.SearchEle();
    iMax.getSearch();
    cout<<"\n Use of Assignment Operator\n";
     iMax2=iMax;
     cout << " \nEnter Element for Searching Int from iMax2\n : " << endl;
     iMax2.SearchEle();
     iMax2.getSearch();
    cout << "\nEnter " << MAX_SIZE << "\nElements for Searching float : " << endl;
    fMax.readElements();
    cout<<"\n Use of Copy constructor\n";
    TClassSearch <float> fCopy = fMax;
    fCopy.SearchEle();
    fCopy.getSearch();
            getch();
        return 0;
}
```

Output:



```
Enter 5
Elements for Searching Int :
1
2
3
4
5

Enter Element to Search : 4

Your Data   :   1       2       3       4       5Class Template Search : Element   : 4 : Found :  Position : 4.

 Use of Assignment Operator

Enter Element for Searching Int from iMax2
 :

Enter Element to Search : 5

Your Data   :   1       2       3       4       5Class Template Search : Element   : 5 : Found :  Position : 5.

Enter 5
Elements for Searching float :
2.2
3.3
4.4
5.5
6.6

 Use of Copy constructor

Enter Element to Search : 3.3
```

Task 7b): Define template functions for compare and use it in the algorithms like bubble sort, insertion sort, merge sort.

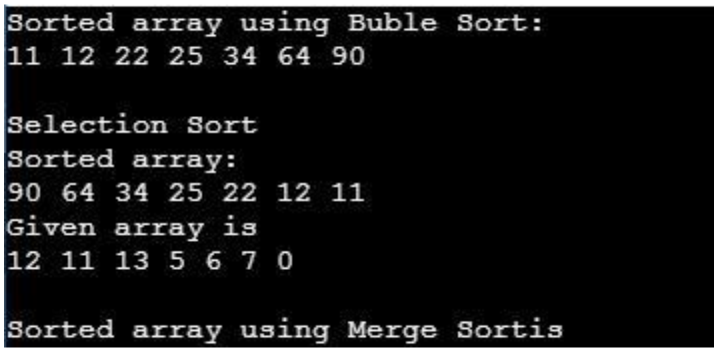
Aim: Write a C++ Program for Define template functions for compare and use it in the algorithms like bubble sort, insertion sort, merge sort.

Program:

```
#include <bits/stdc++.h>
using namespace std;

 template<typename data>
bool compare(data left, data right)
{
   return left > right;
}

void swap(int *xp, int *yp)
{
   int temp = *xp;
   *xp = *yp;
   *yp = temp;
}

// A function to implement bubble sort
void bubbleSort(int arr[], int n)
{
   int i, j;
   for (i = 0; i < n-1; i++)

   // Last i elements are already in place
   for (j = 0; j < n-i-1; j++)
      if (compare(arr[j],arr[j+1]))
         swap(&arr[j], &arr[j+1]);
}

 //selection sort
 void selectionSort(int arr[], int n)
{
   int i, j, min_idx;

   // One by one move boundary of unsorted subarray
   for (i = 0; i < n-1; i++)
   {
      // Find the minimum element in unsorted array
```

```
        min_idx = i;
        for (j = i+1; j < n; j++)
        if (compare(arr[j], arr[min_idx]))
            min_idx = j;

        // Swap the found minimum element with the first element
        swap(&arr[min_idx], &arr[i]);
    }
}
/* Function to print an array */
void printArray(int arr[], int size)
{
    int i;
    for (i = 0; i < size; i++)
        cout << arr[i] << " ";
    cout << endl;
}

//MergeSort
// Merges two subarrays of arr[].
// First subarray is arr[l..m]
// Second subarray is arr[m+1..r]
void merge(int arr[], int l, int m, int r)
{
    int n1 = m - l + 1;
    int n2 = r - m;

    // Create temp arrays
    int L[n1], R[n2];

    // Copy data to temp arrays L[] and R[]
    for (int i = 0; i < n1; i++)
        L[i] = arr[l + i];
    for (int j = 0; j < n2; j++)
        R[j] = arr[m + 1 + j];

    // Merge the temp arrays back into arr[l..r]

    // Initial index of first subarray
    int i = 0;

    // Initial index of second subarray
    int j = 0;

    // Initial index of merged subarray
    int k = l;
    while (i < n1 && j < n2) {
        if (L[i] <= R[j]) {
            arr[k] = L[i];
            i++;
        }
        else {
            arr[k] = R[j];
            j++;
        }
        k++;
    }

    // Copy the remaining elements of
    // L[], if there are any
    while (compare(n1,i)) {
        arr[k] = L[i];
        i++;
        k++;
    }

    // Copy the remaining elements of
    // R[], if there are any
    while (compare(n2,j)) {
        arr[k] = R[j];
        j++;
        k++;
    }
}

// l is for left index and r is
// right index of the sub-array
// of arr to be sorted */
void mergeSort(int arr[],int l,int r){
    if(l>=r){
        return;//returns recursively
    }
    int m =l+ (r-l)/2;
    mergeSort(arr,l,m);
    mergeSort(arr,m+1,r);
    merge(arr,l,m,r);
}
// Driver code
int main()
{
    int arr[] = {64, 34, 25, 12, 22, 11, 90};
    int n = sizeof(arr)/sizeof(arr[0]);
    bubbleSort(arr, n);
    cout<<"Sorted array using Buble Sort: \n";
    printArray(arr, n);
    cout<<"\nSelection Sort\n";
    selectionSort(arr, n);
    cout << "Sorted array: \n";
    printArray(arr, n);
    int arr_merge[] = { 12, 11, 13, 5, 6, 7 };
    int arr_size = sizeof(arr) / sizeof(arr[0]);
    cout << "Given array is \n";
    printArray(arr_merge, arr_size);
    mergeSort(arr_merge, 0, arr_size - 1);
    cout << "\nSorted array using Merge Sortis \n";
    printArray(arr_merge, arr_size);
    return 0;
}
```

Output:



```
Sorted array using Buble Sort:
11 12 22 25 34 64 90

Selection Sort
Sorted array:
90 64 34 25 22 12 11
Given array is
12 11 13 5 6 7 0

Sorted array using Merge Sortis
```

Task-8

Task-8: Formatted input-output examples

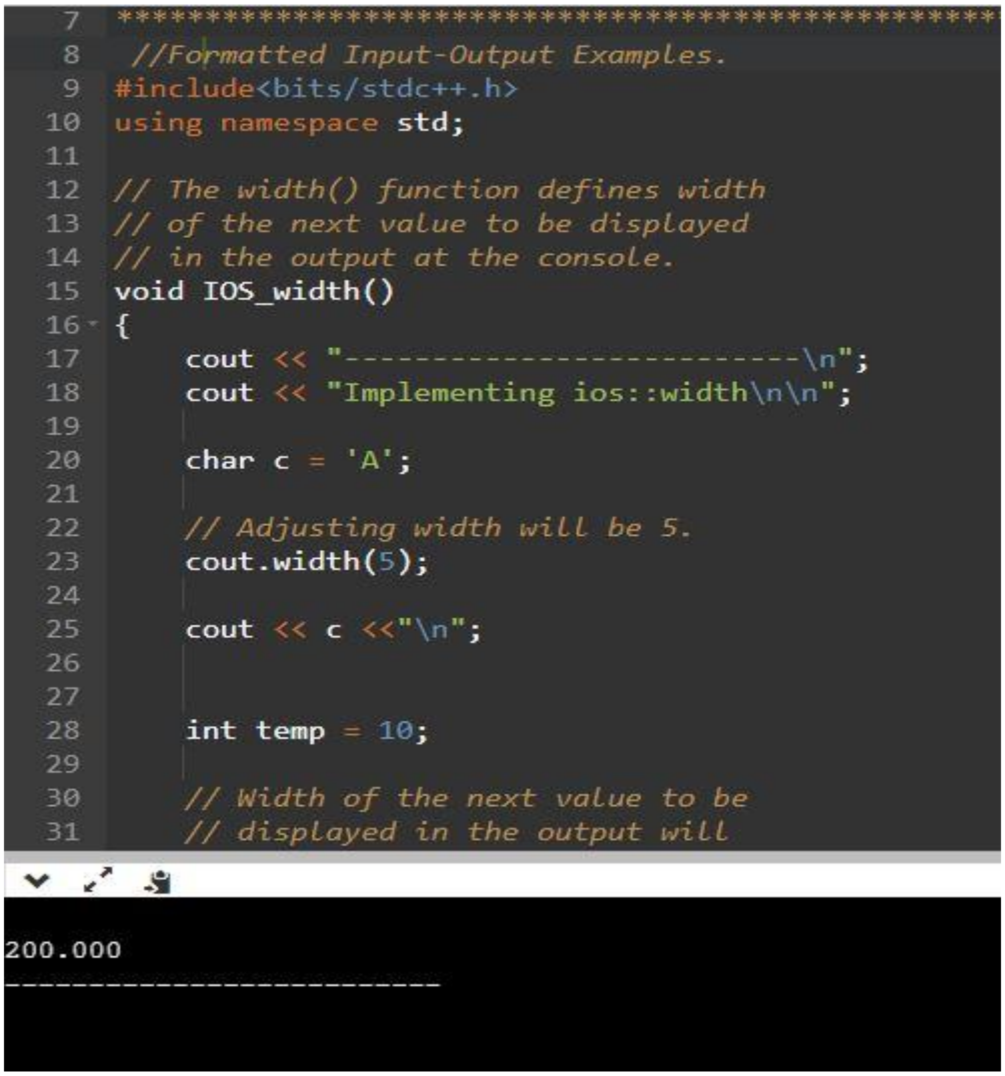
Aim: Write a C++ Program for Formatted input-output examples

Program:

```
#include<bits/stdc++.h>
using namespace std;
  const int SIZE = 100;
// The width() function defines width
// of the next value to be displayed
// in the output at the console.
void IOS_width()
{
    cout << "-------------------------\n";
    cout << "Implementing ios::width\n\n";

    char c = 'A';

    // Adjusting width will be 5.
    cout.width(5);

    cout << c <<"\n";


    int temp = 10;

    // Width of the next value to be
    // displayed in the output will
    // not be adjusted to 5 columns.
    cout<<temp;
    cout << "\n-------------------------\n";
}

void IOS_precision()
{
    cout << "\n-------------------------\n";
    cout << "Implementing ios::precision\n\n";
    cout << "Implementing ios::width";
    cout.setf(ios::fixed, ios::floatfield);
    cout.precision(2);
    cout<<3.1422;
    cout << "\n-------------------------\n";
}
```

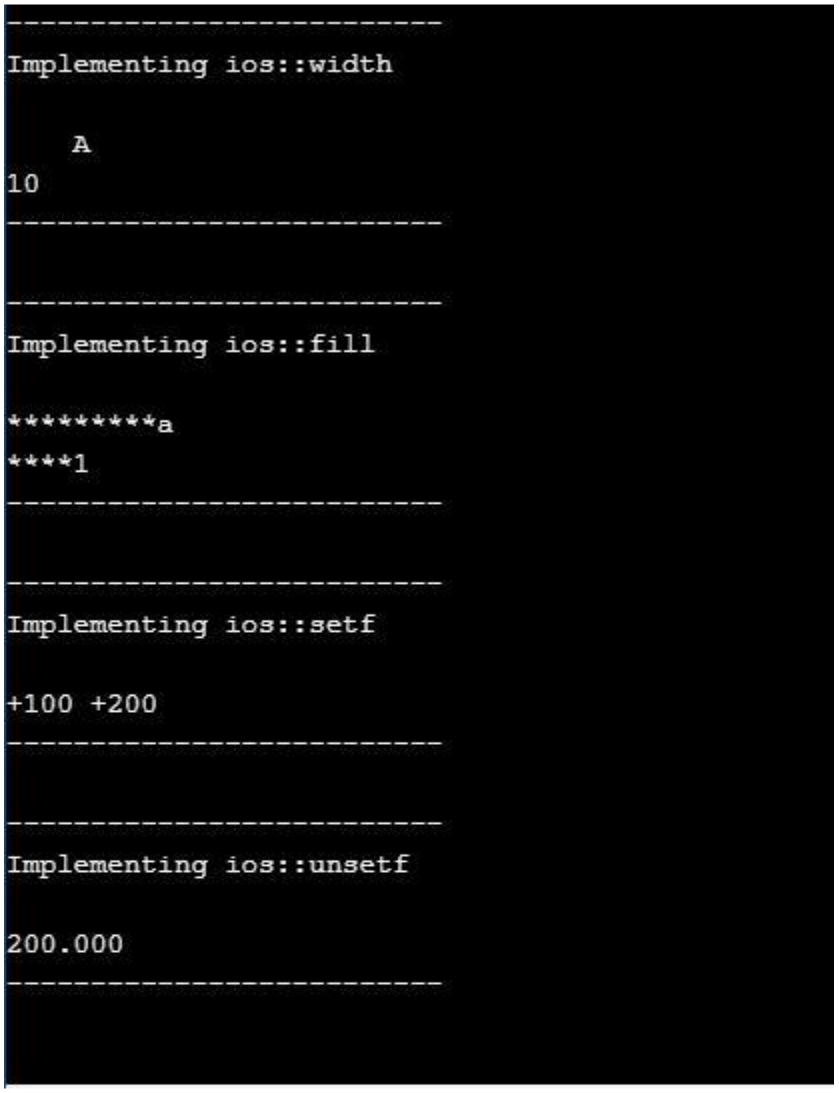
```
// The fill() function fills the unused
// white spaces in a value (to be printed
// at the console), with a character of choice.
void IOS_fill()
{
    cout << "\n--------------------------\n";
    cout << "Implementing ios::fill\n\n";
    char ch = 'a';
    // Calling the fill function to fill
    // the white spaces in a value with a
    // character our of choice.
    cout.fill('*');
    cout.width(10);
    cout<<ch <<"\n";
    int i = 1;
    // Once you call the fill() function,
    // you don't have to call it again to
    // fill the white space in a value with
    // the same character.
    cout.width(5);
    cout<<i;
    cout << "\n--------------------------\n";
}
void IOS_setf()
{
    cout << "\n--------------------------\n";
    cout << "Implementing ios::setf\n\n";
    int val1=100,val2=200;
    cout.setf(ios::showpos);
    cout<<val1<<" "<<val2;
    cout << "\n--------------------------\n";
}
void IOS_unsetf()
{
    cout << "\n--------------------------\n";
    cout << "Implementing ios::unsetf\n\n";
    cout.setf(ios::showpos|ios::showpoint);
    // Clear the showflag flag without
    // affecting the showpoint flag
    cout.unsetf(ios::showpos);
    cout<<200.0;
    cout << "\n--------------------------\n";
}
// Driver Method
int main()
{
    IOS_width();
    IOS_precision;
    IOS_fill();
    IOS_setf();
    IOS_unsetf();
char buffer[SIZE];
cout<<"Enter a line of text:"<<endl;
cin.read(buffer,45);
cout<<"The line of text entered was: "<<endl;
cout.write(buffer, cin.gcount());
// the gcount() member function returns
// the number of unformatted characters last extracted
cout<<endl;
    return 0;
}
```

Output:



```
**************************************************
 //Formatted Input-Output Examples.
#include<bits/stdc++.h>
using namespace std;

// The width() function defines width
// of the next value to be displayed
// in the output at the console.
void IOS_width()
{
    cout << "-------------------------\n";
    cout << "Implementing ios::width\n\n";

    char c = 'A';

    // Adjusting width will be 5.
    cout.width(5);

    cout << c <<"\n";


    int temp = 10;

    // Width of the next value to be
    // displayed in the output will
```

```
200.000
-------------------------
```



```
--------------------------
Implementing ios::width

    A
10
--------------------------


--------------------------
Implementing ios::fill

*********a
****1
--------------------------


--------------------------
Implementing ios::setf

+100 +200
--------------------------


--------------------------
Implementing ios::unsetf

200.000
--------------------------
```

Task-9

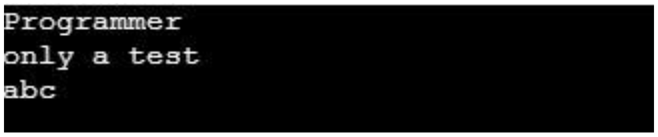
Task-9: Input manipulators

Aim: Write a C++ Program for Input manipulators

Program:

```
//Input Manipulators Examples.
#include <iostream>
#include <istream>
#include <sstream>
#include <string>
using namespace std;
int main()
{
   istringstream str("           Programmer");
   string line;
   // Ignore all the whitespace in string
   // str before the first word.
   getline(str >> std::ws, line);
     // you can also write str>>ws
   // After printing the output it will automatically
   // write a new line in the output stream.
   cout << line << endl;
     // without flush, the output will be the same.
   cout << "only a test" << flush;
     // Use of ends Manipulator
   cout << "\na";
     // NULL character will be added in the Output
   cout << "b" << ends;
   cout << "c" << endl;

   return 0;
}
```

Output:



```
Programmer
only a test
abc
```

Task 10

Task 10: Overriding operators <<, >>

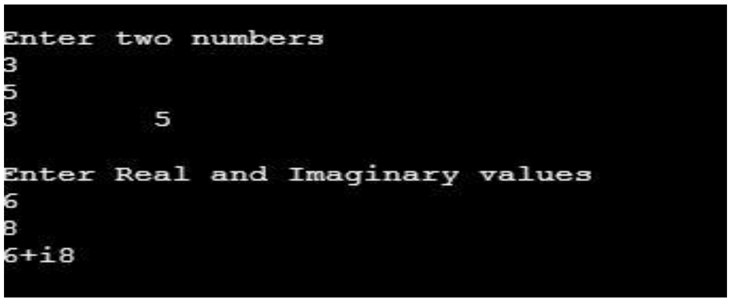
Aim: Write a C++ Program for Overriding operators <<, >>

Program:

```
#include <iostream>
using namespace std;
class numb
{   int a,b;
public:
    virtual void read(istream& is)
{
cout<<"\nEnter two numbers\n";
 cin>>a;
cin>>b;
}
    virtual void print(ostream& os) const
{
cout<<a<<"\t";
cout<<b<<"\n";
 }
};
class complex1: public numb
{   int real,imag;
 public:
    void read(istream& is)
{
cout<<"\nEnter Real and Imaginary values\n";
cin>> real;
 cin>>imag;
}
    void print(ostream& os) const {
cout<<real<<"+i";
cout<<imag;
}
};
istream& operator>>(istream& is, numb& s)
{
s.read (is);
return is;
}
ostream& operator<<(ostream& os, const numb& s)
{
```

```
s.print(os);
return os;
}
istream& operator>>(istream& is, complex1& c1)
{
c1.read (is);
 return is;
}
ostream& operator<<(ostream& os, const complex1& c1)
{
c1.print(os);
 return os;
}
int main( )
    {
        numb *sptr;
        numb s;
 cin>>s;
        cout<<s;
        complex1 c1;
cin>>c1;
        cout<<c1;
         return 0;
    }
```

Output:



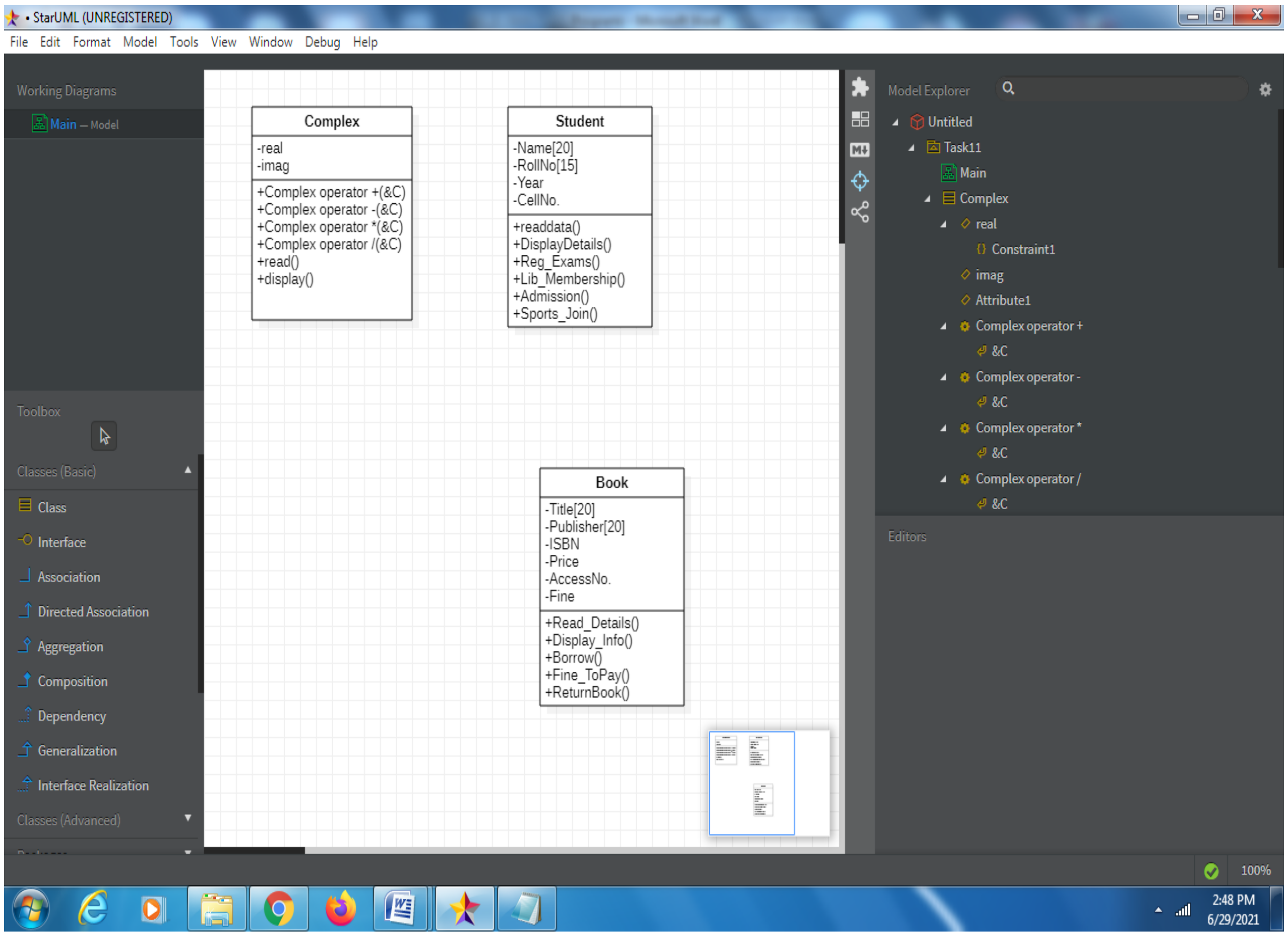
```
Enter two numbers
3
5
3       5

Enter Real and Imaginary values
6
8
6+i8
```

Task 11

Task 11: Define class model for complex number, student class, book class and show it using UML diagram as well as concrete class.

Aim: Write UML Diagrams to Define class model for complex number, student class, book class and show it using UML diagram as well as concrete class.

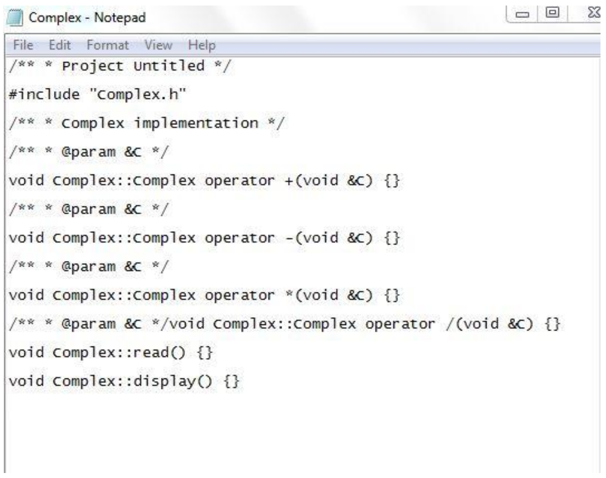
Solution:



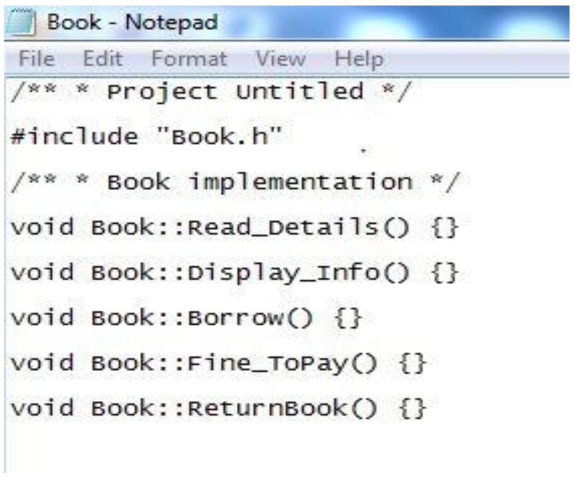
Complex
-real
-imag
+Complex operator +(&C)
+Complex operator -(&C)
+Complex operator *(&C)
+Complex operator /(&C)
+read()
+display()

Student
-Name[20]
-RollNo[15]
-Year
-CellNo.
+readdata()
+DisplayDetails()
+Reg_Exams()
+Lib_Membership()
+Admission()
+Sports_Join()

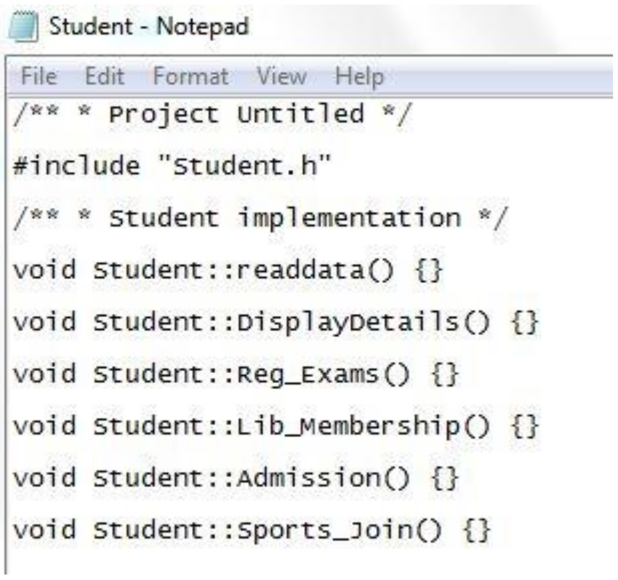
Book
-Title[20]
-Publisher[20]
-ISBN
-Price
-AccessNo.
-Fine
+Read_Details()
+Display_Info()
+Borrow()
+Fine_ToPay()
+ReturnBook()



```
/** * Project Untitled */
#include "Complex.h"
/** * Complex implementation */
/** * @param &C */
void Complex::Complex operator +(void &C) {}
/** * @param &C */
void Complex::Complex operator -(void &C) {}
/** * @param &C */
void Complex::Complex operator *(void &C) {}
/** * @param &C */void Complex::Complex operator /(void &C) {}
void Complex::read() {}
void Complex::display() {}
```



```
/** * Project Untitled */
#include "Book.h"
/** * Book implementation */
void Book::Read_Details() {}
void Book::Display_Info() {}
void Book::Borrow() {}
void Book::Fine_ToPay() {}
void Book::ReturnBook() {}
```



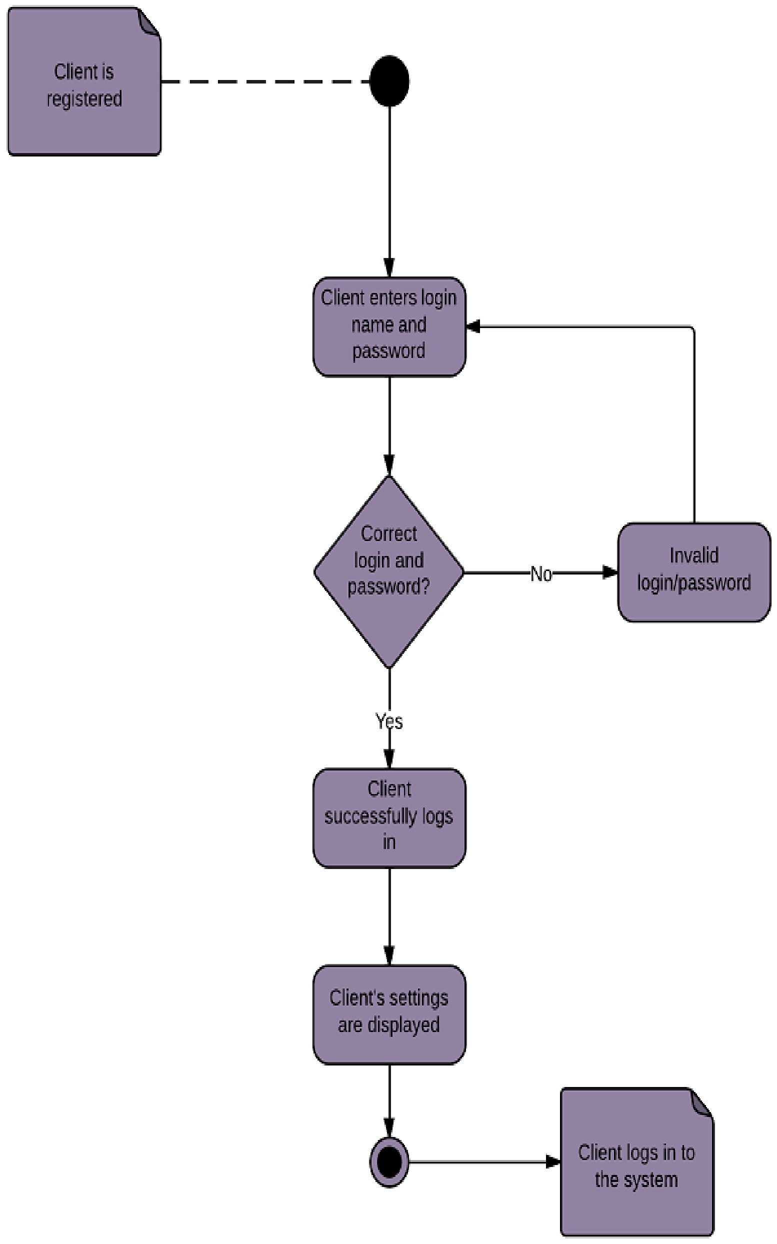
```
/** * Project Untitled */
#include "Student.h"
/** * Student implementation */
void Student::readdata() {}
void Student::DisplayDetails() {}
void Student::Reg_Exams() {}
void Student::Lib_Membership() {}
void Student::Admission() {}
void Student::Sports_Join() {}
```

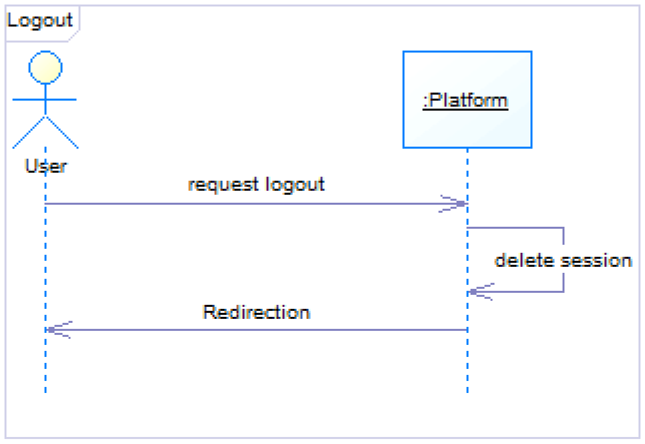
Task-12

Task-12: Show behavioral modeling through sequence diagram and activity diagram for workflow in a typical log-in, log-out situation

Aim: Draw a Diagram to Show behavioral modeling through sequence diagram and activity diagram for workflow in a typical log-in, log-out situation

Solution:.


Client is registered
Client enters login name and password
Correct login and password?
No
Invalid login/password
Yes
Client successfully logs in
Client's settings are displayed
Client logs in to the system


Logout
User
:Platform
request logout
delete session
Redirection