M.Sc. (Five Year Integrated) in Computer Science (Artificial Intelligence & Data Science)

First Semester

Laboratory Record

21-805-0107: C++ PROGRAMMING LAB

Submitted in partial fulfillment
of the requirements for the award of degree in
Master of Science (Five Year Integrated)
in Computer Science (Artificial Intelligence & Data Science) of
Cochin University of Science and Technology (CUSAT)
Kochi

Submitted by

ABDUL HAKKEEM P A
(80521001)

DEPARTMENT OF COMPUTER SCIENCE
COCHIN UNIVERSITY OF SCIENCE AND TECHNOLOGY (CUSAT)
KOCHI-682022

MARCH 2022

DEPARTMENT OF COMPUTER SCIENCE
COCHIN UNIVERSITY OF SCIENCE AND TECHNOLOGY (CUSAT)
KOCHI, KERALA-682022

This is to certify that the software laboratory record for ***21-805-0107: C++ Programming Lab*** *is a record of work carried out by* ***Abdul Hakkeem P A (80521001)****, in partial fulfillment of the requirements for the award of degree in* ***Master of Science (Five Year Integrated)*** *in* ***Computer Science (Artificial Intelligence & Data Science)*** *of Cochin University of Science and Technology (CUSAT), Kochi. The lab record has been approved as it satisfies the academic requirements in respect of the first semester laboratory prescribed for the Master of Science (Five Year Integrated) in Computer Science degree.*

Faculty Member in-charge

Dr. Madhu S. Nair	Dr. Philip Samuel
Professor	Professor and Head
Department of Computer Science	Department of Computer Science
CUSAT	CUSAT

Contents

LEAP YEAR

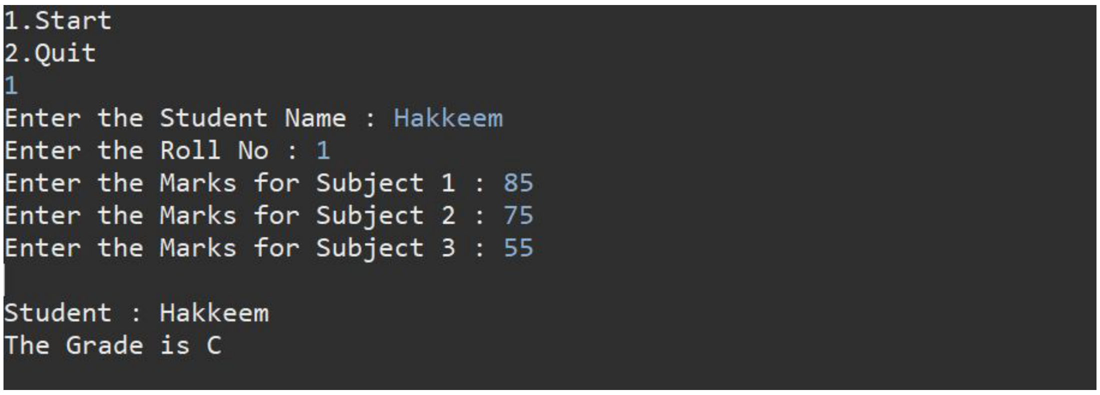
AIM

Write a C++ program to calculate the grades of a list of students with attributes (Name, Roll no, Marks of 3 subjects) using class with member functions input(), calcGrade(), display().

PROGRAM

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

class student{
    std::string name;
    int roll_no;
    float mark1,mark2,mark3,totalMarks;
    char grade;

    public:
        void input();
        char calcGrade(float,float,float);
        void display();
};

void student::input(){
    std::cout<<"Enter the Student Name : ";
    std::cin>>name;
    std::cout<<"Enter the Roll No : ";
    std::cin>>roll_no;
    std::cout<<"Enter the Marks for Subject 1 : ";
    std::cin>>mark1;
    std::cout<<"Enter the Marks for Subject 2 : ";
    std::cin>>mark2;
    std::cout<<"Enter the Marks for Subject 3 : ";
    std::cin>>mark3;
}

char student::calcGrade(float mark1,float mark2,float mark3){
    totalMarks = (mark1+mark2+mark3)/3;
    if(totalMarks>=90){
        grade='A';
    }
    else if(totalMarks>=80 and totalMarks<90){
        grade='B';
    }
    else if(totalMarks>=70 and totalMarks<80){
        grade='C';
    }
    else if(totalMarks>=60 and totalMarks<70){
        grade='D';
    }
    else if(totalMarks>=50 and totalMarks<60){
        grade='E';
    }
    else{
        grade='F';
    }
    return grade;
}

void student::display(){
    std::cout<<endl;
    std::cout<<"Student : "<<name;
    std::cout<<"\nThe Grade is "<<calcGrade(mark1,mark2,mark3)<<std::endl;
}

int main() {
    int exitOption,choice;
    std::cout<<"1.Start\n2.Quit"<<endl;
    std::cin>>exitOption;
    if (exitOption==2) {
        return 0;
    }
    do{
        student Student1;
        Student1.input();
        Student1.display();
        std::cout<<"\nDo you want to continue \n1.Continue\n2.Quit"<<std::endl;
        std::cin>>choice;
    } while(choice == 1);
    return 0;
}
```

SAMPLE INPUT-OUTPUT



```
1.Start
2.Quit
1
Enter the Student Name : Hakkeem
Enter the Roll No : 1
Enter the Marks for Subject 1 : 85
Enter the Marks for Subject 2 : 75
Enter the Marks for Subject 3 : 55

Student : Hakkeem
The Grade is C
```

PALINDROME NUMBER

AIM

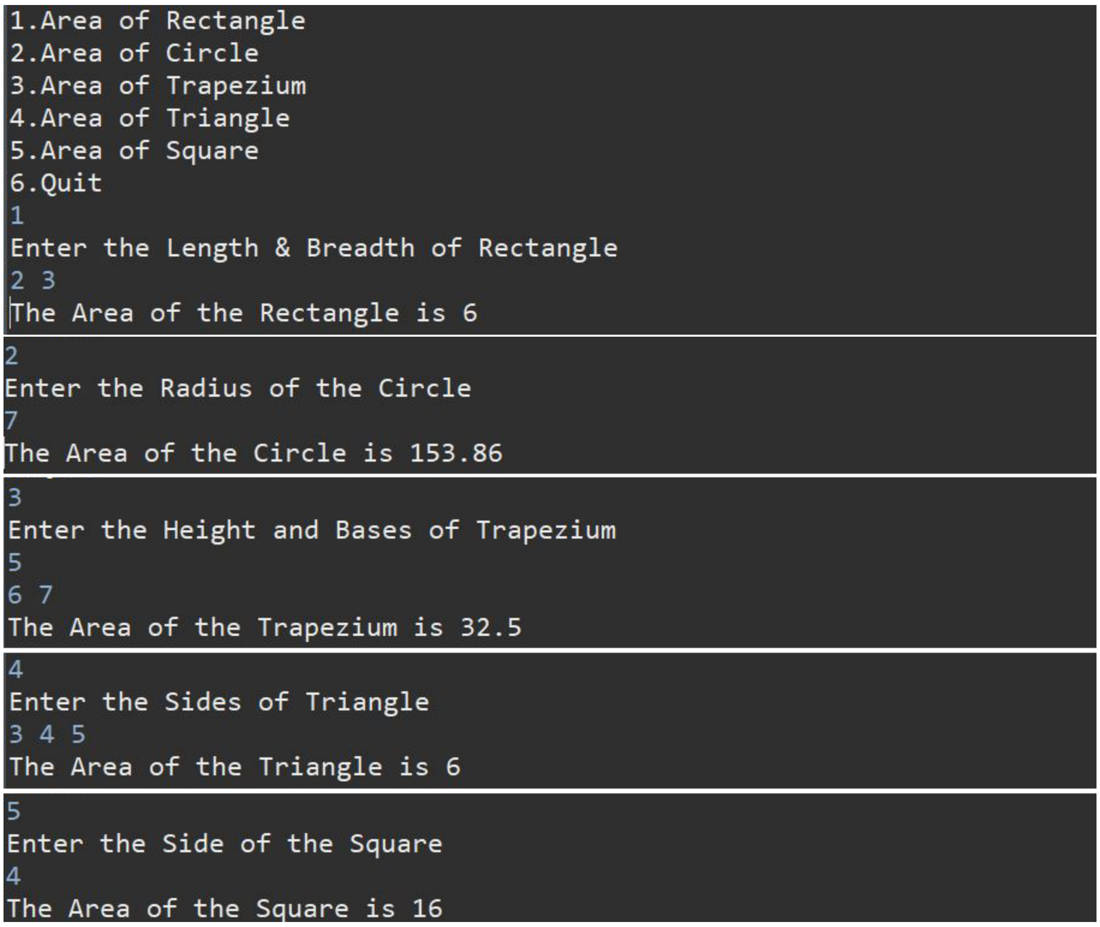
Write a C++ prgram to calculate the area of different shapes like Rectangle, Square etc (at least 5 shapes) using overloaded function area().

PROGRAM

```
#include <iostream>
#include <cmath>

void area(int length,int breadth){
    int area;
    area=length*breadth;
    std::cout<<"The Area of the Rectangle is "<<area<<std::endl;
}

void area(int side){
    int area;
    area=side*side;
    std::cout<<"The Area of the Square is "<<area<<std::endl;
}

void area(float radius){
    float area;
    area=3.14*radius*radius;
    std::cout<<"The Area of the Circle is "<<area<<std::endl;
}

void area(float height,float base1,float base2){
    float area;
    area=0.5*(height*(base1+base2));
    std::cout<<"The Area of the Trapezium is "<<area<<std::endl;
}

void area(int side1,int side2,int side3){
    float area;
    float s=(side1+side2+side3)/2;
    area=std::sqrt(s*(s-side1)*(s-side2)*(s-side3));
    std::cout<<"The Area of the Triangle is "<<area<<std::endl;
}
```

```
int main() {
    int choice,option;
    do{
        std::cout<<"1.Area of Rectangle\n2.Area of Circle\n3.Area of
        Trapezium\n4.Area of Triangle\n5.Area of Square\n6.Quit"<<std::endl;
        std::cin>>choice;
        switch(choice){
                case 1:
                    int l,b;
                    std::cout<<"Enter the Length & Breadth of Rectangle\n";
                    std::cin>>l>>b;
                    area(l, b);
                    break;
                case 2:
                    float radius;
                    std::cout<<"\n\nEnter the Radius of the Circle\n";
                    std::cin>>radius;
                    area(radius);
                    break;
                case 3:
                    float height,base1,base2;
                    std::cout<<"\n\nEnter the Height and Bases of Trapezium\n";
                    std::cin>>height>>base1>>base2;
                    area(height, base1, base2);
                    break;
                case 4:
                    int side1,side2,side3;
                    std::cout<<"\n\nEnter the Sides of Triangle\n";
                    std::cin>>side1>>side2>>side3;
                    area(side1, side2, side3);
                    break;
                case 5:
                    int side;
                    std::cout<<"\n\nEnter the Side of the Square\n";
                    std::cin>>side;
                    area(side);
                    break;
                case 6:
                    break;
        }
        std::cout<<"Do you want to continue?\n1.Continue\n2.Quit\n";
        std::cin>>option;
        }
        while(option == 1);
        return 0;
    }
```

SAMPLE INPUT-OUTPUT



```
1.Area of Rectangle
2.Area of Circle
3.Area of Trapezium
4.Area of Triangle
5.Area of Square
6.Quit
1
Enter the Length & Breadth of Rectangle
2 3
The Area of the Rectangle is 6
```

```
2
Enter the Radius of the Circle
7
The Area of the Circle is 153.86
```

```
3
Enter the Height and Bases of Trapezium
5
6 7
The Area of the Trapezium is 32.5
```

```
4
Enter the Sides of Triangle
3 4 5
The Area of the Triangle is 6
```

```
5
Enter the Side of the Square
4
The Area of the Square is 16
```

LEAP YEAR

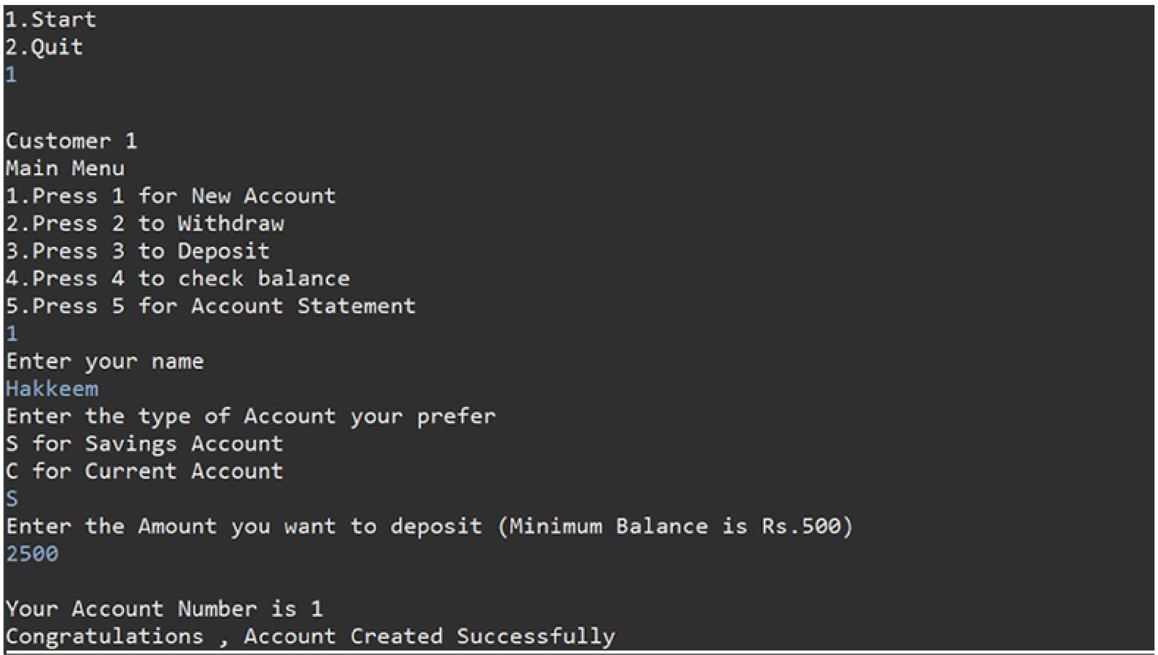
AIM

Write a C++ program using classes to perform bank transaction for n customers (cust name, acc no, acc type, balance). The program should be menu driven and it should have the following menus like adding new account, withdraw (keep a min balance of 500), deposit, balance enquiry and account statement (cust name, acc no, acc type, balance).

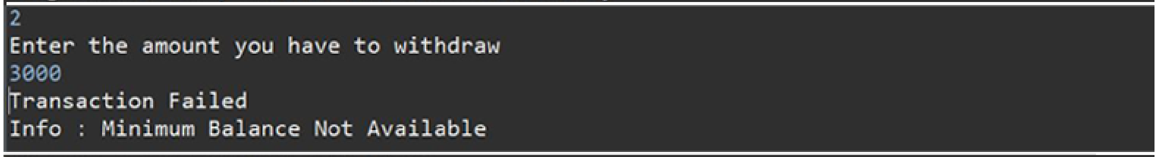
PROGRAM

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

class bank{
    string cust_name;
    int acc_no;
    char acc_type;
    float balance,customer_amount;
    int static count;
public:
    void new_acc(void);
    void withdraw(void);
    void deposit(void);
    void balance_enquiry(void);
    void acc_statement(void);
};
int bank::count;

void bank::new_acc(){
    count++;
    acc_no=count;
    cin.ignore(numeric_limits<streamsize>::max(),'\n');
    cout<<"Enter your name"<<endl;
    getline(cin,cust_name);
    cout<<"Enter the type of Account your prefer\nS for Savings
    Account\nC for Current Account\n";
    cin>>acc_type;
    cout<<"Enter the Amount you want to deposit (Minimum Balance is Rs.500)\n";
    cin>>balance;
    cout<<"\nYour Account Number is "<<acc_no<<endl;
    cout<<"Congratulations , Account Created Successfully\n";
}

void bank::withdraw(){
    cout<<"Enter the amount you have to withdraw\n";
    cin>>customer_amount;
    if ((balance-customer_amount)>500) {
        balance=balance-customer_amount;
        cout<<"Rs."<<customer_amount<<" is withdrawn\n";
    } else {
        cout<<"Transaction Failed\nInfo : Minimum Balance Not Available\n";
    }
}

void bank::deposit(){
    cout<<"Enter the amount you have to deposit\n";
    cin>>customer_amount;
    balance=customer_amount+balance;
    cout<<"Rs."<<customer_amount<<" is deposited\n";
}

void bank::balance_enquiry(){
    cout<<"Current Balance is Rs."<<balance<<endl;
}

void bank::acc_statement(){
    cout<<"------------------------------------\n";
    cout<<"Your Account Statement is \n";
    cout<<"Account Holder  : "<<cust_name<<endl;
    cout<<"Account No      : "<<acc_no<<endl;
    cout<<"Account Type    : "<<acc_type<<endl;
    cout<<"Account Balance : Rs."<<balance<<endl;
    cout<<"------------------------------------\n";
}

int main() {
    const int customer_number=20;
    int choice;
    bool Close = false;
    bank* customer=new bank[customer_number];
    int leave,exitOption;
    cout<<"1.Start\n2.Quit"<<endl;
    cin>>exitOption;
    if (exitOption==2) {
        return 0;
    }
    for (int i = 0; i < customer_number; ++i) {
        if(Close){
            leave = 3;
        }
        else{
            leave = 1;
        }
        while (leave == 1){
            cout<<"\nCustomer "<<i+1<<endl;
            cout<<"Main Menu\n1.Press 1 for New Account\n2.Press 2 to
            Withdraw\n3.Press 3 to Deposit\n4.Press 4 to check balance\n
            5.Press 5 for Account Statement\n";
            cin>>choice;
            switch (choice) {
                case 1:
                    customer[i].new_acc();
                    break;
                case 2:
                    customer[i].withdraw();
                    break;
                case 3:
                    customer[i].deposit();
                    break;
                case 4:
                    customer[i].balance_enquiry();
                    break;
                case 5:
                    customer[i].acc_statement();
                    break;
                default:
                    cout<<"Error! , Try Again"<<endl;
                    break;
            }
            cout<<"\nDo you want to continue or quit.\nPress 1 to Continue
            \nPress 2 to Next Customer\nPress 3 to Quit the Application\n";
            cin>>leave;
            if (leave==3) {
                Close = true;
            }
        }
    }
    return 0;
}
```

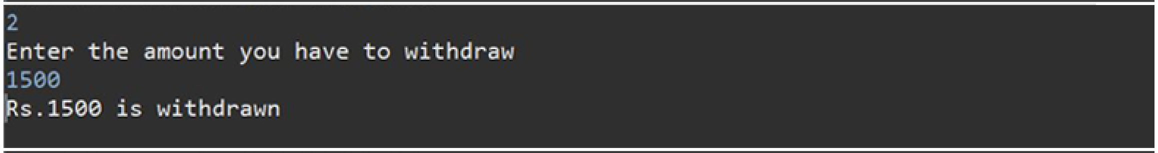
SAMPLE INPUT-OUTPUT



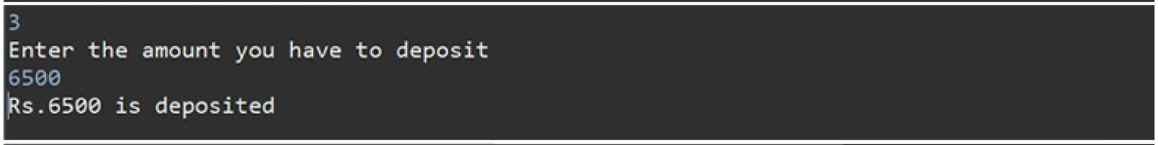
```
1.Start
2.Quit
1


Customer 1
Main Menu
1.Press 1 for New Account
2.Press 2 to Withdraw
3.Press 3 to Deposit
4.Press 4 to check balance
5.Press 5 for Account Statement
1
Enter your name
Hakkeem
Enter the type of Account your prefer
S for Savings Account
C for Current Account
S
Enter the Amount you want to deposit (Minimum Balance is Rs.500)
2500

Your Account Number is 1
Congratulations , Account Created Successfully
```



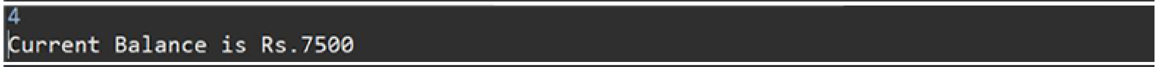
```
2
Enter the amount you have to withdraw
3000
Transaction Failed
Info : Minimum Balance Not Available
```



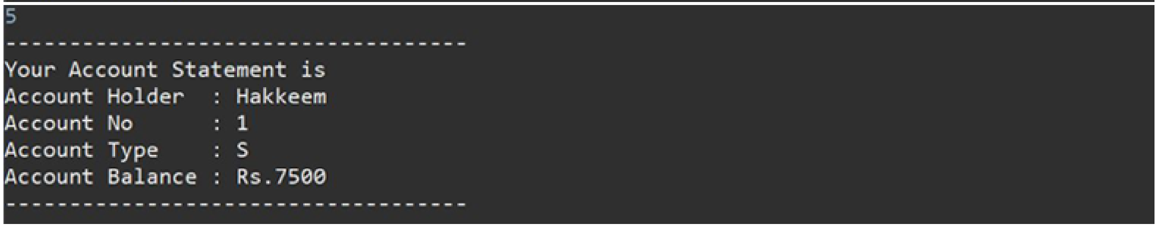
```
2
Enter the amount you have to withdraw
1500
Rs.1500 is withdrawn
```



```
3
Enter the amount you have to deposit
6500
Rs.6500 is deposited
```



```
4
Current Balance is Rs.7500
```



```
5
-----------------------------------
Your Account Statement is
Account Holder  : Hakkeem
Account No      : 1
Account Type    : S
Account Balance : Rs.7500
-----------------------------------
```

LEAP YEAR

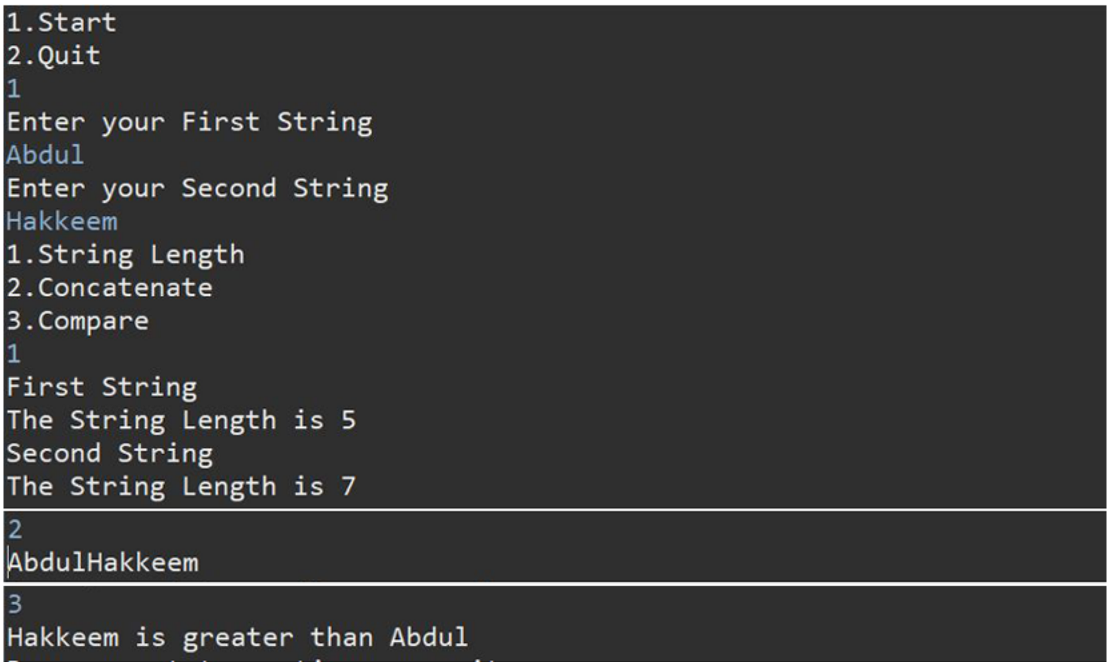
AIM

Write a C++ progtram to perform operations such as compare, concatenate and length on String objects.

PROGRAM

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

class Strings{
    char *name;
    int length;
public:
    Strings(){
        length=0;
        name = new char[length+1];
    }
    ~Strings(){
        delete name;
    }
    void getString(char *txt);
    void compare(const Strings&,const Strings&);
    void concatenate(const Strings&,const Strings&);
    void displayLength();
};

void Strings::getString(char *txt){
    length = strlen(txt);
    delete name;
    name = new char[length+1];
    strcpy(name, txt);
}

void Strings::concatenate(const Strings &string1,const Strings &string2){
    length = string1.length + string2.length;
    delete name;
    name = new char[length+1];
    strcpy(name, string1.name);
    strcat(name, string2.name);
    cout<<name<<endl;
}

void Strings::compare(const Strings &string1,const Strings &string2){
    if (string1.length == string2.length) {
        cout<<"They are same strings"<<endl;
    }
    else if (string1.length > string2.length) {
        cout<<string1.name<<" is greater than "<<string2.name<<endl;
    }
    else {
        cout<<string2.name<<" is greater than "<<string1.name<<endl;
    }
}

void Strings::displayLength(){
    cout<<"The String Length is "<<length<<endl;
}

int main() {
    char name1[50],name2[50];
    Strings string1,string2,result;
    int choice,option,exitOption;
    cout<<"1.Start\n2.Quit"<<endl;
    cin>>exitOption;
    if (exitOption==2) {
        return 0;
    }
    cout<<"Enter your First String"<<endl;
    cin>>name1;
    cout<<"Enter your Second String"<<endl;
    cin>>name2;
    string1.getString(name1);
    string2.getString(name2);
    do
    {
        cout<<"1.String Length\n2.Concatenate\n3.Compare"<<endl;
    cin>>choice;
    switch (choice){
        case 1:
                cout<<"First String"<<endl;
                string1.displayLength();
                cout<<"Second String"<<endl;
                string2.displayLength();
                break;
            case 2:
                result.concatenate(string1, string2);
                break;
            case 3:
                result.compare(string1,string2);
                break;
            default:
                cout<<"Invalid Choice"<<endl;
                break;
        }
        cout<<"Do you want to continue or quit\n1.Continue\n2.Quit"<<endl;
        cin>>option;
    } while (option == 1);
    if(option != 1){
        cout<<"Succesfully Quitted"<<endl;
    }
    return 0;
}
```

SAMPLE INPUT-OUTPUT



```
1.Start
2.Quit
1
Enter your First String
Abdul
Enter your Second String
Hakkeem
1.String Length
2.Concatenate
3.Compare
1
First String
The String Length is 5
Second String
The String Length is 7
```

```
2
AbdulHakkeem
```

```
3
Hakkeem is greater than Abdul
```

LEAP YEAR

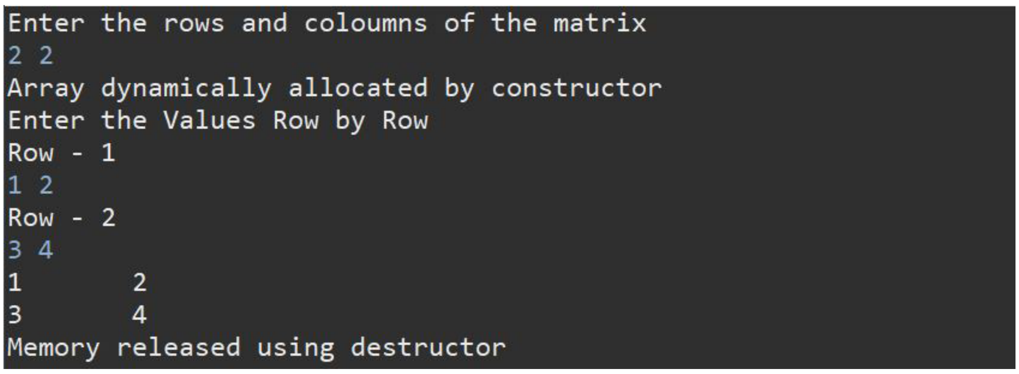
AIM

Write a C++ program to demonstrate the order of execution of constructors & destructors.

PROGRAM

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

class A{
    int **p;
    int d1,d2;
public:
    A(int x,int y){
        cout<<"Array dynamically allocated by constructor"<<endl;
        d1 = x;
        d2 = y;
        p = new int *[d1];
        for (int i = 0; i < d1; ++i) {
            p[i] = new int [d2];
        }
    }
    void get_data();
    void put_data();
    ~A(){
        cout<<"Memory released using destructor"<<endl;
    }
};

void A::get_data(){
    cout<<"Enter the Values Row by Row"<<endl;
    for (int i = 0; i < d1; ++i) {
        cout<<"Row - "<<i+1<<endl;
        for (int j = 0; j < d2; ++j) {
            cin>>p[i][j];
        }
    }
}

void A::put_data(){
    for (int i = 0; i < d1; ++i) {
        for (int j = 0; j < d2; ++j) {
            cout<<p[i][j]<<"\t";
        }
        cout<<endl;
    }
}

int main() {
    int row,col;
    cout<<"Enter the rows and coloumns of the matrix"<<endl;
    cin>>row>>col;
    A Object(row,col);
    Object.get_data();
    Object.put_data();
    return 0;
}
```

SAMPLE INPUT-OUTPUT



```
Enter the rows and coloumns of the matrix
2 2
Array dynamically allocated by constructor
Enter the Values Row by Row
Row - 1
1 2
Row - 2
3 4
1       2
3       4
Memory released using destructor
```

LEAP YEAR

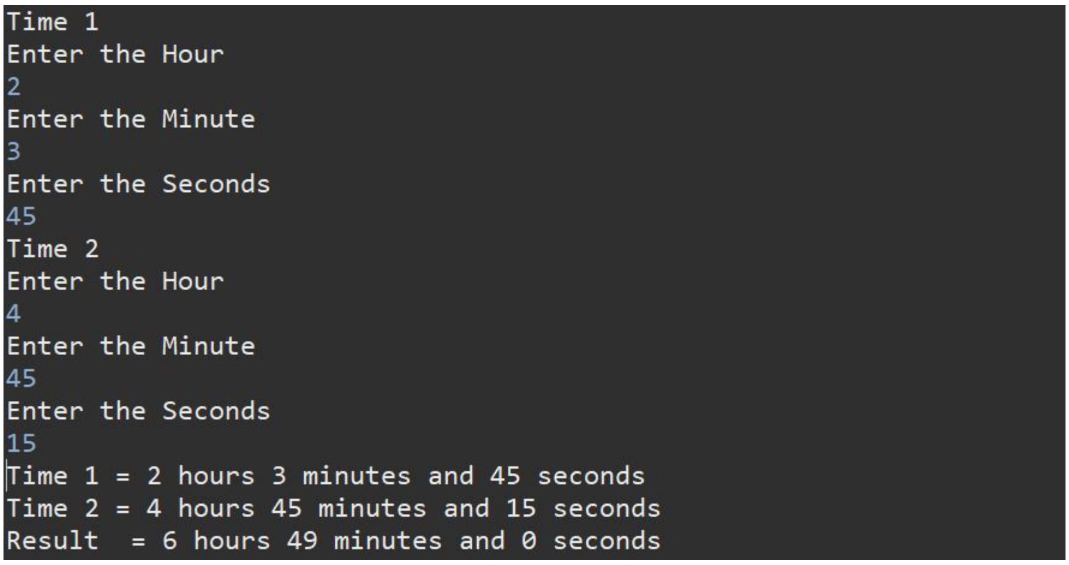
AIM

Create a class TIME with members hours, minutes, seconds. Take input, add two time objects by passing objects to function and display result.

PROGRAM

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

class Time{
    int hours,minutes,seconds;
public:
    void gettime(){
        cout<<"Enter the Hour\n";
        cin>>hours;
        cout<<"Enter the Minute\n";
        cin>>minutes;
        cout<<"Enter the Seconds\n";
        cin>>seconds;
    }
    void puttime(void){
        cout<<hours<<" hours "<<minutes<<" minutes and "<<seconds<<" seconds"<<endl;
    }
    void sum(Time,Time);
};

void Time::sum(Time t1,Time t2){
    seconds = t1.seconds+t2.seconds;
    minutes = seconds/60;
    seconds = seconds%60;
    minutes = minutes+t1.minutes+t2.minutes;
    hours=minutes/60;
    minutes=minutes%60;
    hours=hours+t1.hours+t2.hours;
}

int main() {
    Time T1,T2,T3;
    cout<<"Time 1"<<endl;
    T1.gettime();
    cout<<"Time 2"<<endl;
    T2.gettime();
    T3.sum(T1, T2);
    cout<<"Time 1 = ";
    T1.puttime();
    cout<<"Time 2 = ";
    T2.puttime();
    cout<<"Result  = ";
    T3.puttime();
    return 0;
}
```

SAMPLE INPUT-OUTPUT



```
Time 1
Enter the Hour
2
Enter the Minute
3
Enter the Seconds
45
Time 2
Enter the Hour
4
Enter the Minute
45
Enter the Seconds
15
Time 1 = 2 hours 3 minutes and 45 seconds
Time 2 = 4 hours 45 minutes and 15 seconds
Result  = 6 hours 49 minutes and 0 seconds
```

LEAP YEAR

AIM

Write a C++ program to implement a class MATRIX with member functions such as matrix_add, matrix_mult, matrix_transpose, matrix_determinant etc.

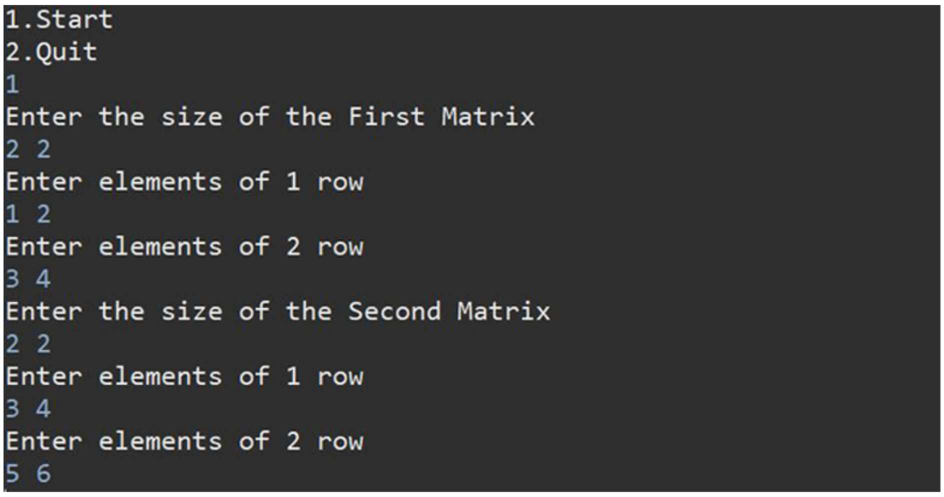
PROGRAM

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

class matrices{
    int rows,coloumns;
    int **matrix;
public:
    matrices(int r,int c){
        rows = r;
        coloumns = c;
        matrix = new int *[rows];
        for (int i = 0; i < rows; ++i) {
            matrix[i]= new int [coloumns];
        }
    }
    ~matrices(){
        for (int i = 0; i < rows; ++i) {
            delete matrix[i];
        }
        delete matrix;
        cout<<"Memory Released"<<endl;
    }
    matrices(){};
    void get_matrix();
    friend void matrix_add(const matrices&,const matrices&);
    friend void matrix_mult(const matrices&,const matrices&);
    void matrix_transpose();
    void matrix_trace();
};

void matrices::get_matrix(){
    for (int i = 0; i < rows; ++i) {
        cout<<"Enter elements of "<<i+1<<" row"<<endl;
        for (int j = 0; j < coloumns; ++j) {
            cin>>matrix[i][j];
        }
    }
}
void matrix_add(const matrices &a,const matrices &b){
    if(a.rows==b.rows and a.coloumns==b.coloumns){
        int sum[a.rows][a.coloumns];
        for (int i = 0; i < a.rows; ++i) {
            for (int j = 0; j < a.coloumns; ++j) {
                sum[i][j]=a.matrix[i][j]+b.matrix[i][j];
            }
        }
        for (int i = 0; i < a.rows; ++i) {
            for (int j = 0; j < b.coloumns; ++j) {
                cout<<sum[i][j]<<"\t";
            }
            cout<<endl;
        }
    }
    else{
        cout<<"Addition not possible"<<endl;
    }
}

void matrices::matrix_transpose(){
    int transpose[coloumns][rows];
    //to take the transpose
    for (int i = 0; i < rows; ++i) {
        for (int j = 0; j < coloumns; ++j) {
            transpose[j][i] = matrix[i][j];
        }
    }
    //to display the transpose
    cout<<"Transpose of the Matrix"<<endl;
    for (int i = 0; i < coloumns; ++i) {
        for (int j = 0; j < rows; ++j) {
            cout<<transpose[i][j]<<"\t";
        }
        cout<<"\n";
    }
```

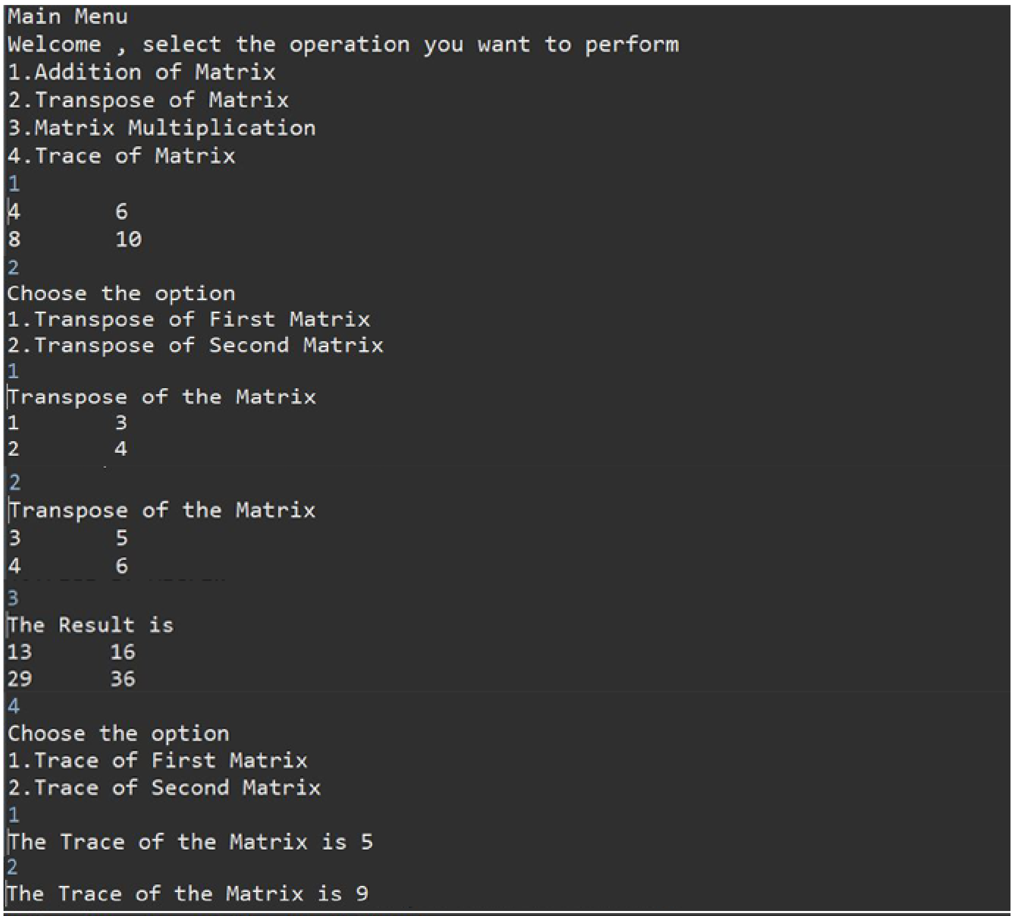
```
}

void matrix_mult(const matrices &a,const matrices &b){
    if (a.coloumns==b.rows) {
        int mult[a.rows][b.coloumns];
        for (int i = 0; i < a.rows; ++i) {
            for (int j = 0; j < b.coloumns; ++j) {
                int sum = 0;
                for (int k = 0; k < a.coloumns; ++k) {
                    sum = sum + a.matrix[i][k]*b.matrix[k][j];
                    mult[i][j]=sum;
                }
            }
        }
        //printing result
        cout<<"The Result is"<<endl;
        for (int i = 0; i < a.rows; ++i) {
            for (int j = 0; j < b.coloumns; ++j) {
                cout<<mult[i][j]<<"\t";
            }
            cout<<endl;
        }
    } else {
        cout<<"For Matrix Multiplication , no of coloumns of first
        matrix should be equal to no of rows of second matrix"<<endl;
    }
}

void matrices::matrix_trace(){
    if (rows !=coloumns){
        cout<<"Trace is possible for square matrix"<<endl;
    }
    else{
        int trace = 0;
        for (int i = 0; i < rows; ++i) {
            trace = trace + matrix[i][i];
        }
        cout<<"The Trace of the Matrix is "<<trace<<endl;
    }
}
```

```
int main() {
    int Exit;
    cout<<"1.Start\n2.Quit"<<endl;
    cin>>Exit;
    if (Exit==2) {
        return 0;
    }
    int choice,m,n,opt,exit_option;
    cout<<"Enter the size of the First Matrix"<<endl;
    cin>>m>>n;
    matrices matrix1(m,n);
    matrix1.get_matrix();
    cout<<"Enter the size of the Second Matrix"<<endl;
    cin>>m>>n;
    matrices matrix2(m,n);
    matrix2.get_matrix();
    do {
        cout<<"Main Menu"<<endl;
        cout<<"Welcome , select the operation you want to perform"<<endl;
        cout<<"1.Addition of Matrix\n2.Transpose of Matrix\n3.Matrix
        Multiplication\n4.Trace of Matrix"<<endl;
        cin>>choice;
        switch (choice) {
            case 1:
                matrix_add(matrix1, matrix2);
                break;
            case 2:
                cout<<"Choose the option\n1.Transpose of First Matrix\n2.
                Transpose of Second Matrix"<<endl;
                cin>>opt;
                if(opt == 1){
                    matrix1.matrix_transpose();
                }
                else {
                    matrix2.matrix_transpose();
                }
                break;
            case 3:
                matrix_mult(matrix1, matrix2);
                break;
            case 4:
                cout<<"Choose the option\n1.Trace of First Matrix\n2.Trace of
                Second Matrix"<<endl;
                cin>>opt;
                if(opt == 1){
                    matrix1.matrix_trace();
                }
                else {
                    matrix2.matrix_trace();
                }
                break;
            default:
                cout<<"Invalid Choice";
                break;
        }
        cout<<"Do you want to continue?\n1.Continue\n2.Exit"<<endl;
        cin>>exit_option;
    } while (exit_option==1);
    if (exit_option!=1){
        cout<<"Succesfully Exited"<<endl;
    }
    return 0;
}
```

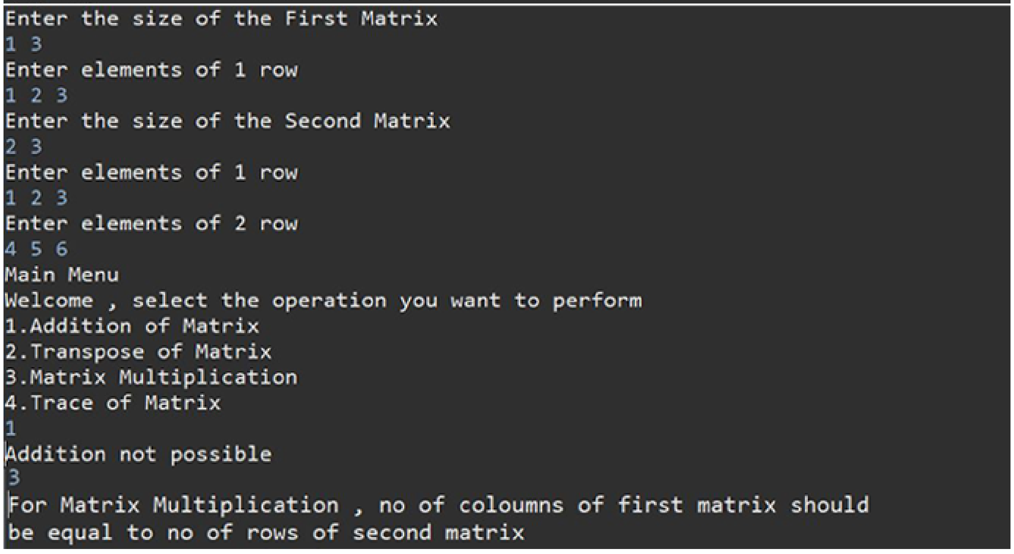
SAMPLE INPUT-OUTPUT



```
1.Start
2.Quit
1
Enter the size of the First Matrix
2 2
Enter elements of 1 row
1 2
Enter elements of 2 row
3 4
Enter the size of the Second Matrix
2 2
Enter elements of 1 row
3 4
Enter elements of 2 row
5 6
```



```
Main Menu
Welcome , select the operation you want to perform
1.Addition of Matrix
2.Transpose of Matrix
3.Matrix Multiplication
4.Trace of Matrix
1
4       6
8       10
2
Choose the option
1.Transpose of First Matrix
2.Transpose of Second Matrix
1
Transpose of the Matrix
1       3
2       4
2
Transpose of the Matrix
3       5
4       6
3
The Result is
13      16
29      36
4
Choose the option
1.Trace of First Matrix
2.Trace of Second Matrix
1
The Trace of the Matrix is 5
2
The Trace of the Matrix is 9
```



```
Enter the size of the First Matrix
1 3
Enter elements of 1 row
1 2 3
Enter the size of the Second Matrix
2 3
Enter elements of 1 row
1 2 3
Enter elements of 2 row
4 5 6
Main Menu
Welcome , select the operation you want to perform
1.Addition of Matrix
2.Transpose of Matrix
3.Matrix Multiplication
4.Trace of Matrix
1
Addition not possible
3
For Matrix Multiplication , no of coloumns of first matrix should
be equal to no of rows of second matrix
```

LEAP YEAR

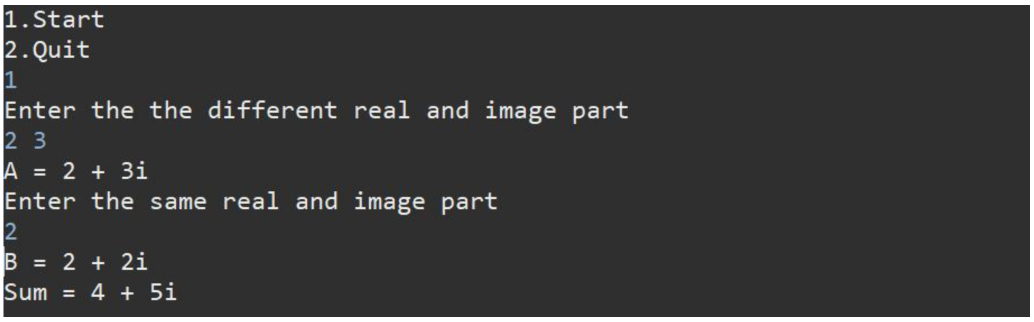
AIM

Write a program to perform addition of two complex numbers using constructor overloading. The first constructor which takes no argument is used to create objects which are not initialized, second which takes one argument is used to initialize real and imag parts to equal values and third which takes two argument is used to initialized real and imag to two different values.

PROGRAM

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

class complex{
    float real,image;
public:
    complex(){}
    complex(float a){
        real=image=a;
    }
    complex(float x,float y){
        real=x;
        image=y;
    }
    friend complex sum(complex,complex);
    friend void display(complex);
};

complex sum(complex A,complex B){
    complex result;
    result.real=A.real+B.real;
    result.image=A.image+B.image;
    return result;
}

void display(complex number){
    if (number.image<0) {
        cout<<number.real<<" "<<number.image<<"i"<<endl;
    } else {
        cout<<number.real<<" + "<<number.image<<"i"<<endl;
    }
}

int main() {
    int exitOption,loopOption;
    cout<<"1.Start\n2.Quit"<<endl;
    cin>>exitOption;
    if (exitOption==2) {
        return 0;
    }
    do {
        float num1,num2;
        cout<<"Enter the different real and image part\n";
        cin>>num1>>num2;
        complex A(num1,num2);
        cout<<"A = ";
        display(A);
        float num3;
        cout<<"Enter the same real and image part"<<endl;
        cin>>num3;
        complex B(num3);
        cout<<"B = ";
        display(B);
        complex C;
        C=sum(A,B);
        cout<<"Sum = ";
        display(C);
        cout<<"Do you want to continue ?\n1.Continue\n2.Quit"<<endl;
        cin>>loopOption;
    } while (loopOption==1);
    return 0;
}
```

SAMPLE INPUT-OUTPUT



```
1.Start
2.Quit
1
Enter the the different real and image part
2 3
A = 2 + 3i
Enter the same real and image part
2
B = 2 + 2i
Sum = 4 + 5i
```

LEAP YEAR

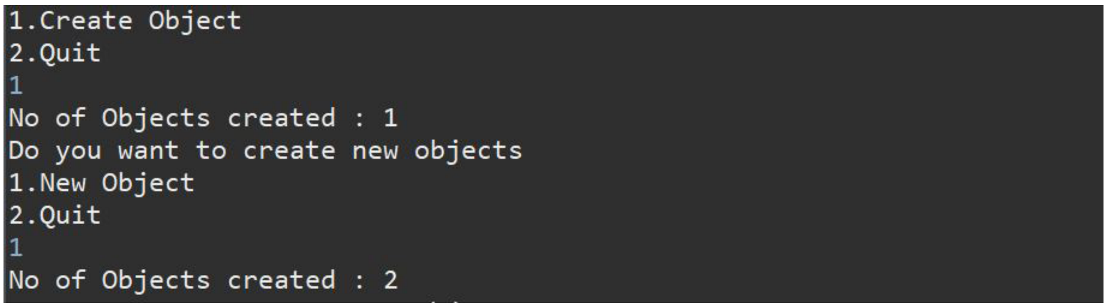
AIM

Write a C++ program to design a class having static member function named showcount() which has the property of displaying the number of objects created of the class.

PROGRAM

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

class object{
    int static count;
public:
    object(){
        count++;
    }
    void static showcount(){
        cout<<"No of Objects created : "<<count<<endl;
    }
};

int object::count;

int main() {
    int choice,exitOption;
    cout<<"1.Create Object\n2.Quit"<<endl;
    cin>>exitOption;
    if(exitOption==2){
        return 0;
    }
    do {
        object *a = new object;
        a->showcount();
        delete a;
        cout<<"Do you want to create new objects \n1.New Object \n2.Quit"<<endl;
        cin>>choice;
    } while (choice==1);
    return 0;

}
```

SAMPLE INPUT-OUTPUT



```
1.Create Object
2.Quit
1
No of Objects created : 1
Do you want to create new objects
1.New Object
2.Quit
1
No of Objects created : 2
```

LEAP YEAR

AIM

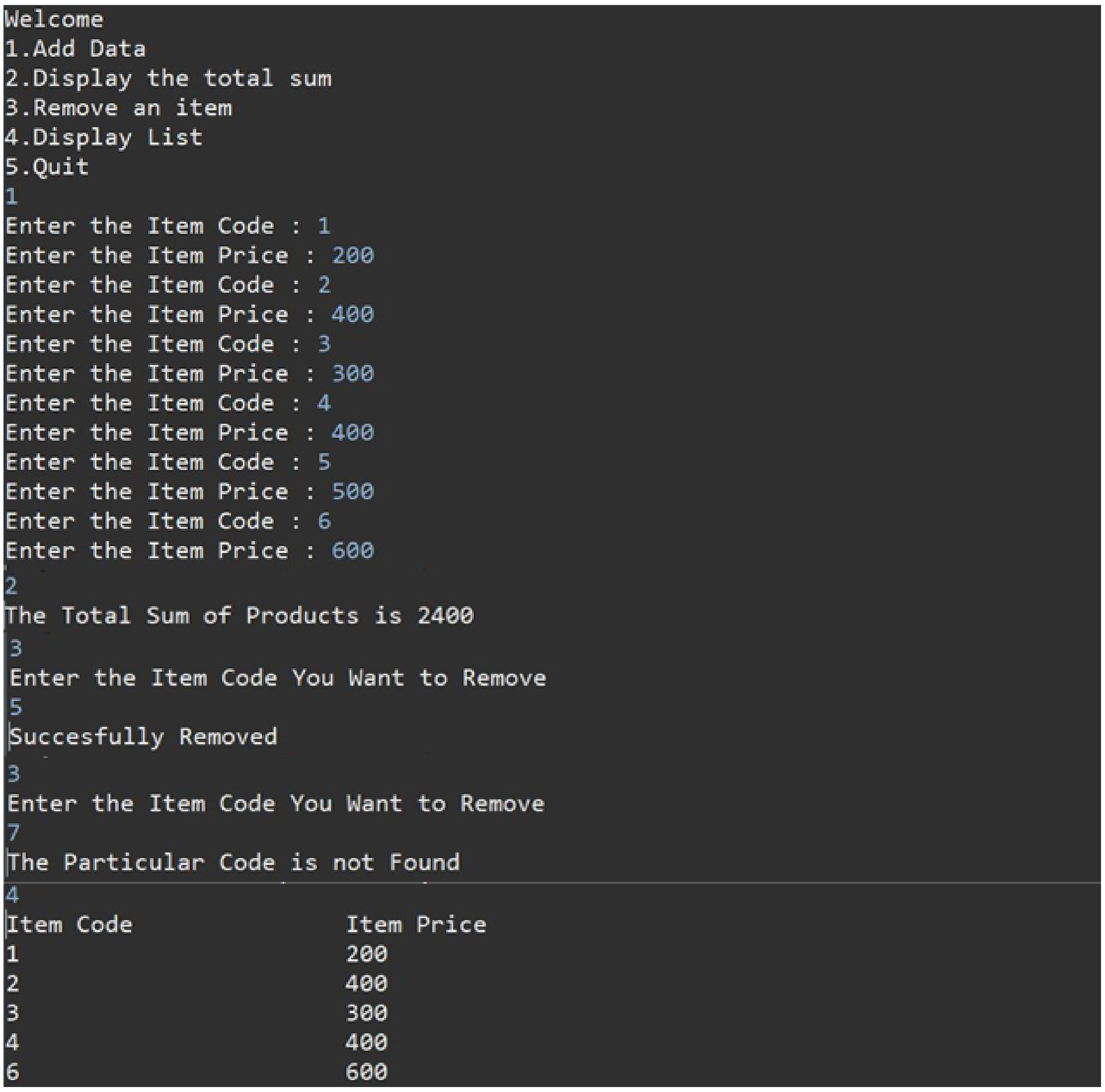
Write a C++ program using class to process shopping list for a Departmental Store. The list include details such as the Code-no and price of each item and perform the operations like adding deleting items to the list and printing the total value of an order.

PROGRAM

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

const int size=6;
class list{
    int itemcode[size];
    float itemprice[size];
    float sum;
    static int count;
    bool removed;
    int n_removed;

public:
    list(){
        n_removed = 0;
    }
    void getdata();
    void displaysum();
    void remove();
    void displaylist();
};

void list::getdata(){
    for (int i = 0; i < size; ++i) {
        cout<<"Enter the Item Code : ";
        cin>>itemcode[i];
        cout<<"Enter the Item Price : ";
        cin>>itemprice[i];
        count++;
    }
}
void list::displaysum(){
    sum=0;
    for (int i = 0; i < size; ++i) {
        sum=sum+itemprice[i];
    }
    cout<<"The Total Sum of Products is "<<sum<<endl;
}
void list::remove(){
    int itemCode;
    cout<<"Enter the Item Code You Want to Remove\n";
    cin>>itemCode;
    removed=false;
    int position;
    for (int i = 0; i < size; ++i) {
        if (itemCode==itemcode[i]) {
            position=i;
            removed=true;
        }
    }
    if (removed) {
        cout<<"Succesfully Removed "<<endl;
        n_removed++;
        for (int i = position; i < size; ++i) {
            itemcode[i]=itemcode[i+1];
            itemprice[i]=itemprice[i+1];
        }
    }
    else {
        cout<<"The Particular Code is not Found"<<endl;
    }
}

void list::displaylist(){
    cout<<"Item Code\t\t";
    cout<<"Item Price"<<endl;
    if(removed){
        for (int i = 0; i < size-n_removed; ++i) {
        cout<<itemcode[i]<<"\t\t\t";
        cout<<itemprice[i]<<endl;
        }
    }
    else{
        for (int i = 0; i < size; ++i) {
            cout<<itemcode[i]<<"\t\t\t";
            cout<<itemprice[i]<<endl;
        }
    }
}

int list::count;

int main() {
    list stock1;
    int choice,option;
    do {
        cout<<"Welcome\n1.Add Data\n2.Display the total sum\n3.Remove an
        item\n4.Display List\n5.Quit"<<endl;
        cin>>choice;
        switch (choice)
        {
            case 1:
                stock1.getdata();
                break;
            case 2:
                stock1.displaysum();
                break;
            case 3:
                stock1.remove();
                break;
            case 4:
                stock1.displaylist();
                break;
            case 5:
                return 0;
                break;
            default:
                cout<<"Invalid Choice"<<endl;
                break;
        }
        cout<<"Do you want to continue or quit\n1.Continue\n2.Quit"<<endl;
        cin>>option;
    } while (option==1);
    return 0;
```

```
}
```

SAMPLE INPUT-OUTPUT



```
Welcome
1.Add Data
2.Display the total sum
3.Remove an item
4.Display List
5.Quit
1
Enter the Item Code : 1
Enter the Item Price : 200
Enter the Item Code : 2
Enter the Item Price : 400
Enter the Item Code : 3
Enter the Item Price : 300
Enter the Item Code : 4
Enter the Item Price : 400
Enter the Item Code : 5
Enter the Item Price : 500
Enter the Item Code : 6
Enter the Item Price : 600
2
The Total Sum of Products is 2400
3
Enter the Item Code You Want to Remove
5
Succesfully Removed
3
Enter the Item Code You Want to Remove
7
The Particular Code is not Found
4
Item Code               Item Price
1                       200
2                       400
3                       300
4                       400
6                       600
```

LEAP YEAR

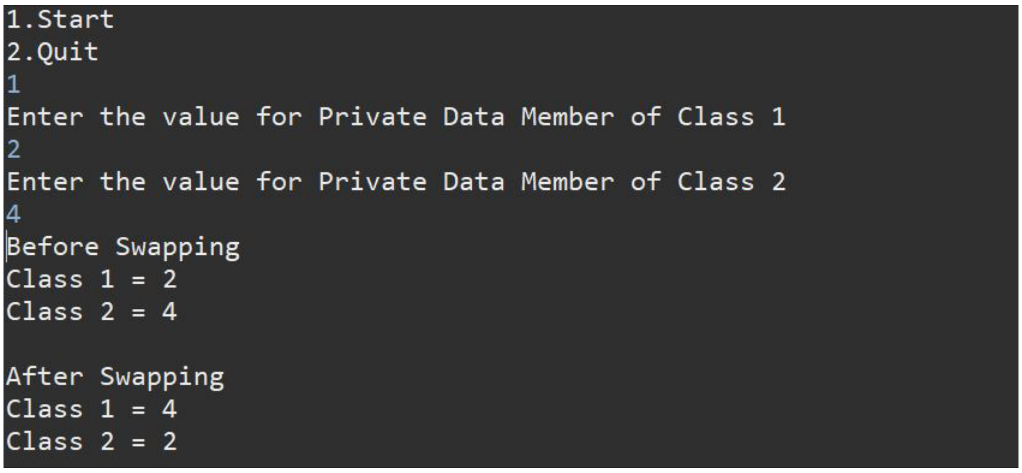
AIM

Write a Program to swap private data members of classes named as class_1, class_2 using friend function.

PROGRAM

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

class class_2;
class class_1{
    int num1;
public:
    class_1(){};
    void get_value_firstClass(){
        cout<<"Enter the value for Private Data Member of Class 1"<<endl;
        cin>>num1;
    }
    friend void swap(class_1&,class_2&);
    void display_class1(){
        cout<<"Class 1 = "<<num1<<endl;
    }
};

class class_2{
    int num2;
public:
    class_2(){};
    void get_value_secClass(){
        cout<<"Enter the value for Private Data Member of Class 2"<<endl;
        cin>>num2;
    }
    friend void swap(class_1&,class_2&);
    void display_class2(){
        cout<<"Class 2 = "<<num2<<endl;
    }
};

void swap(class_1 &a,class_2 &b){
    int temp;
    temp=a.num1;
    a.num1=b.num2;
    b.num2=temp;
}

int main() {
    int exitOption;
    cout<<"1.Start\n2.Quit"<<endl;
    cin>>exitOption;
    if (exitOption==2) {
        return 0;
    }
    class_1 obj1;
    class_2 obj2;
    obj1.get_value_firstClass();
    obj2.get_value_secClass();
    cout<<"Before Swapping"<<endl;
    obj1.display_class1();
    obj2.display_class2();
    swap(obj1, obj2);
    cout<<"\nAfter Swapping"<<endl;
    obj1.display_class1();
    obj2.display_class2();
    return 0;
}
```

SAMPLE INPUT-OUTPUT



```
1.Start
2.Quit
1
Enter the value for Private Data Member of Class 1
2
Enter the value for Private Data Member of Class 2
4
Before Swapping
Class 1 = 2
Class 2 = 4

After Swapping
Class 1 = 4
Class 2 = 2
```

LEAP YEAR

AIM

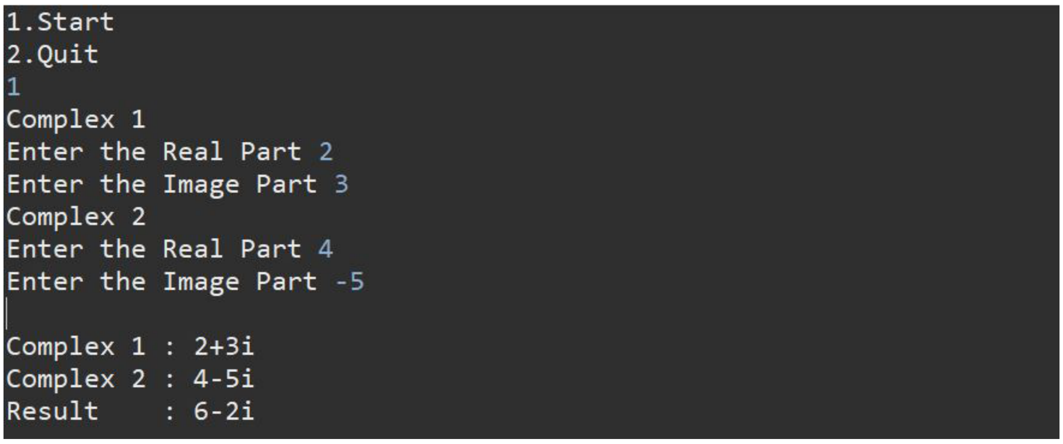
Write a Program to design a class complex to represent complex numbers. The complex class should use an external function (use it as a friend function) to add two complex numbers. The function should return an object of type complex representing the sum of two complex numbers.

PROGRAM

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

class complex{
    int real,image;
public:
    void getvalue();
    complex friend sum(complex,complex);
    void display();
};
void complex::getvalue(){
    cout<<"Enter the Real Part ";
    cin>>real;
    cout<<"Enter the Image Part ";
    cin>>image;
}

void complex::display(){
    if (image<0) {
        cout<<real<<image<<"i"<<endl;
    } else {
        cout<<real<<"+"<<image<<"i"<<endl;
    }
}

complex sum(complex a,complex b){
    complex result;
    result.real=a.real+b.real;
    result.image=a.image+b.image;
    return result;
}
```

```
int main() {
    int exitOption;
    cout<<"1.Start\n2.Quit"<<endl;
    cin>>exitOption;
    if (exitOption==2) {
        return 0;
    }
    complex a,b,result;
    cout<<"Complex 1"<<endl;
    a.getvalue();
    cout<<"Complex 2"<<endl;
    b.getvalue();
    cout<<"\nComplex 1 : ";
    a.display();
    cout<<"Complex 2 : ";
    b.display();
    result=sum(a, b);
    cout<<"Result    : ";
    result.display();
    return 0;
}
```

SAMPLE INPUT-OUTPUT



```
1.Start
2.Quit
1
Complex 1
Enter the Real Part 2
Enter the Image Part 3
Complex 2
Enter the Real Part 4
Enter the Image Part -5

Complex 1 : 2+3i
Complex 2 : 4-5i
Result    : 6-2i
```

LEAP YEAR

AIM

Write a C++ program to overload ==, ! =, <, <=, > and >= operators as member operator functions for a vector object.

PROGRAM

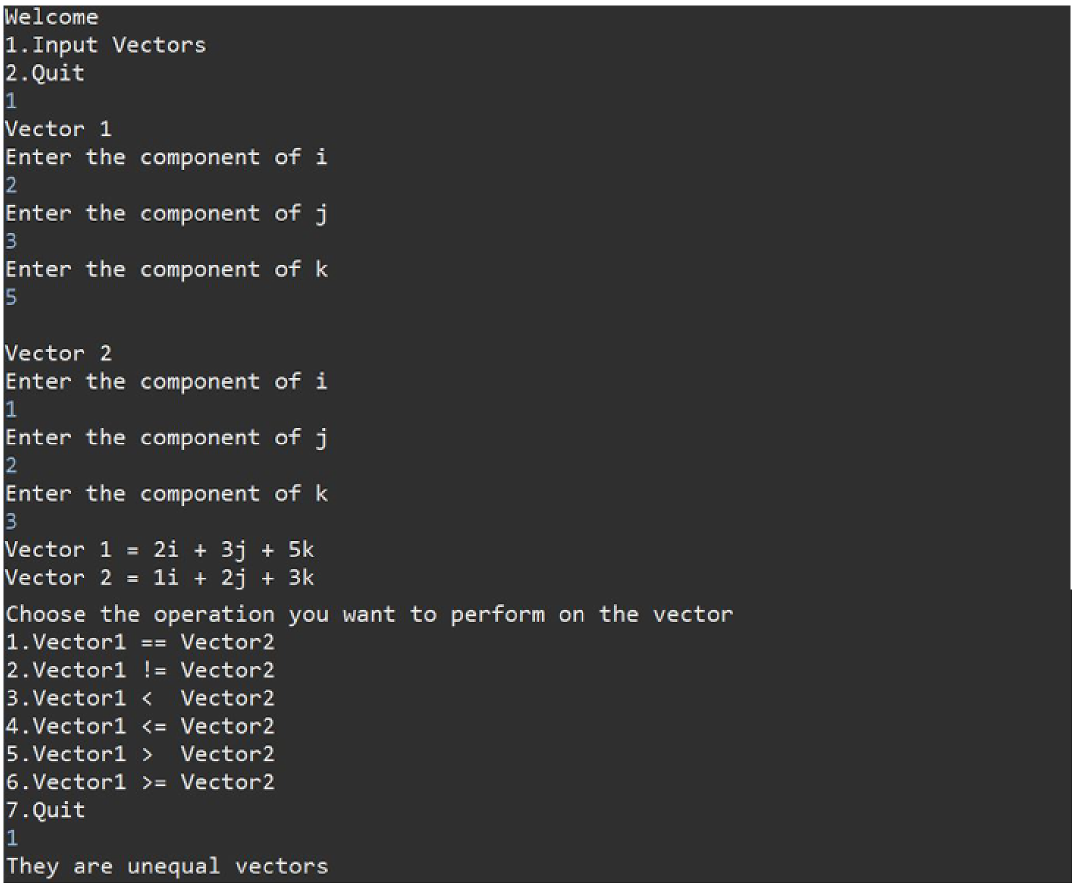
```
#include <iostream>
#include <cmath>
using namespace std;

class Vector{
float i_Component,j_Component,k_Component,magnitude;
public:
void getVector();
void displayVector();
void FindMagnitude();
void operator==(const Vector&);
void operator!=(const Vector&);
void operator<(const Vector&);
void operator<=(const Vector&);
void operator>(const Vector&);
void operator>=(const Vector&);
};

//member function to get the components of the vector.
void Vector::getVector(){
cout<<"Enter the component of i"<<endl;
cin>>i_Component;
cout<<"Enter the component of j"<<endl;
cin>>j_Component;
cout<<"Enter the component of k"<<endl;
cin>>k_Component;
}

//member function to display the vector entered.
void Vector::displayVector(){
if (j_Component>=0 and k_Component>=0) {
cout<<i_Component<<"i + "<<j_Component<<"j + "<<k_Component<<"k"<<endl;
}
```

```
else {
if(j_Component<0 and k_Component<0){
cout<<i_Component<<"i "<<j_Component<<"j "<<k_Component<<"k"<<endl;
}
else if(j_Component<0){
cout<<i_Component<<"i "<<j_Component<<"j + "<<k_Component<<"k"<<endl;
}
else {
cout<<i_Component<<"i + "<<j_Component<<"j "<<k_Component<<"k"<<endl;
}
}
}

//member function to compute the magnitude of the vector
void Vector::FindMagnitude(){
magnitude = (i_Component*i_Component)+(j_Component*j_Component)+(k_Component*k_Component);
magnitude = sqrtf(magnitude);
//cout<<"Magnitude is"<<magnitude<<endl;
}

//member function to overload == operator for vector
void Vector::operator==(const Vector &A){
if (magnitude == A.magnitude){
cout<<"They are equal vectors"<<endl;
}
else {
cout<<"They are unequal vectors"<<endl;
}
}
//member function to overload != operator for vector
void Vector::operator!=(const Vector &A){
if (magnitude != A.magnitude){
cout<<"They are unequal vectors"<<endl;
}
else {
cout<<"They are equal vectors"<<endl;
}
}
//member function to overload < operator for vector
void Vector::operator<(const Vector &A){
if (magnitude < A.magnitude) {
```

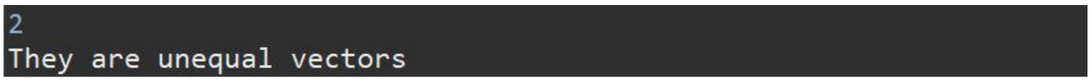
```
cout<<"Vector 2 is greater than Vector 1"<<endl;
}
else {
cout<<"Vector 1 is greater than Vector 2"<<endl;
}
}
//member function to overload <= operator for vector
void Vector::operator<=(const Vector &A){
if (magnitude < A.magnitude) {
cout<<"Vector 2 is greater than Vector 1"<<endl;
}
else if(magnitude == A.magnitude){
cout<<"They are equal vectors"<<endl;
}
else {
cout<<"Vector 1 is greater than Vector 2"<<endl;
}
}
//member function to overload > operator for vector
void Vector::operator>(const Vector &A){
if (magnitude > A.magnitude) {
cout<<"Vector 1 is greater than Vector 2"<<endl;
}
else {
cout<<"Vector 2 is greater than Vector 1"<<endl;
}
}
//member function to overload >= operator for vector
void Vector::operator>=(const Vector &A){
if (magnitude > A.magnitude) {
cout<<"Vector 1 is greater than Vector 2"<<endl;
}
else if(magnitude == A.magnitude){
cout<<"They are equal vectors"<<endl;
}
else {
cout<<"Vector 2 is greater than Vector 1"<<endl;
}
}

int main() {
```

```
int option;
cout<<"Welcome\n1.Input Vectors\n2.Quit"<<endl;
cin>>option;

if (option==2) {
cout<<"You have successfully quited"<<endl;
return 0;
}

Vector V1,V2;
int choice,choice2;
cout<<"Vector 1"<<endl;
V1.getVector();
cout<<"\nVector 2"<<endl;
V2.getVector();
cout<<"Vector 1 = ";
V1.displayVector();
cout<<"Vector 2 = ";
V2.displayVector();
V1.FindMagnitude();
V2.FindMagnitude();

do {
cout<<"\nChoose the operation you want to perform on the vector \n1.Vector1 == Vector2\n2.
cin>>choice;
switch (choice) {
case 1:
V1== V2;
break;
case 2:
V1!= V2;
break;
case 3:
V1< V2;
break;
case 4:
V1<= V2;
break;
case 5:
V1> V2;
break;
case 6:
V1>= V2;
break;
default:
break;
}
cout<<"Do you want to Continue ?\n1.Continue\n2.Quit"<<endl;
cin>>choice2;
} while (choice2 == 1);
if(choice2!=1){
cout<<"Successfully Quitted!"<<endl;
}
return 0;
}
```

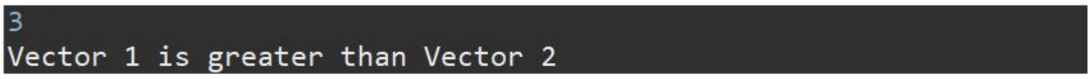
SAMPLE INPUT-OUTPUT



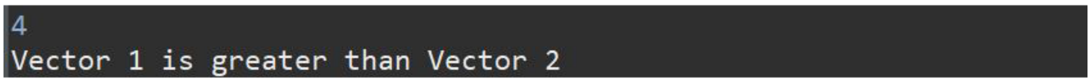
```
Welcome
1.Input Vectors
2.Quit
1
Vector 1
Enter the component of i
2
Enter the component of j
3
Enter the component of k
5

Vector 2
Enter the component of i
1
Enter the component of j
2
Enter the component of k
3
Vector 1 = 2i + 3j + 5k
Vector 2 = 1i + 2j + 3k
Choose the operation you want to perform on the vector
1.Vector1 == Vector2
2.Vector1 != Vector2
3.Vector1 <  Vector2
4.Vector1 <= Vector2
5.Vector1 >  Vector2
6.Vector1 >= Vector2
7.Quit
1
They are unequal vectors
```



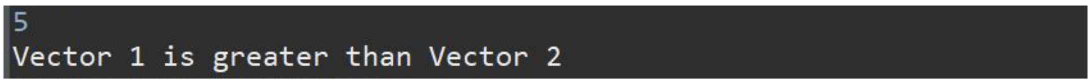
```
2
They are unequal vectors
```



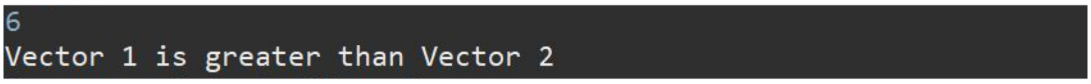
```
3
Vector 1 is greater than Vector 2
```



```
4
Vector 1 is greater than Vector 2
```



```
5
Vector 1 is greater than Vector 2
```



```
6
Vector 1 is greater than Vector 2
```

LEAP YEAR

AIM

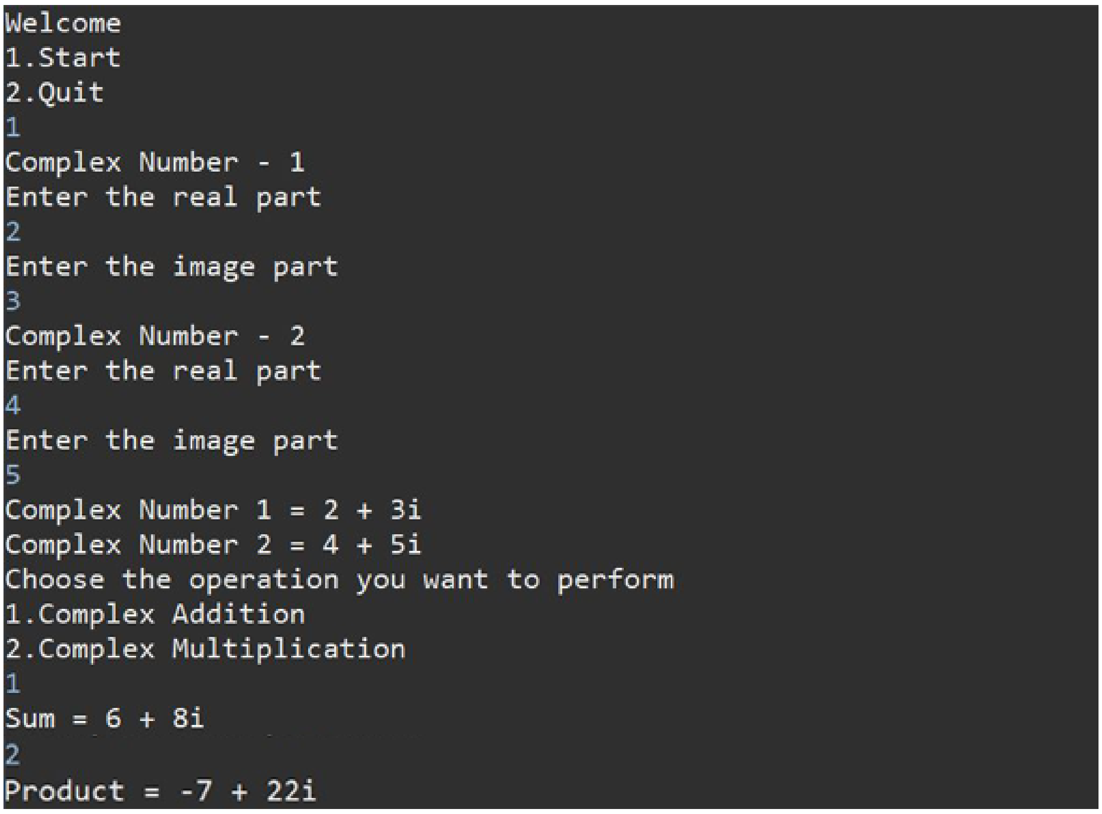
Write a C++ program to design a class representing complex numbers and having the functionality of performing addition & multiplication of two complex numbers using operator overloading (Use friend operator functions).

PROGRAM

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

class Complex{
float real,image;
public:
void getComplex();
void displayComplex();
friend Complex operator+(const Complex&,const Complex&);
friend Complex operator*(const Complex&,const Complex&);
};
//member function to get the complex number.
void Complex::getComplex(){
cout<<"Enter the real part"<<endl;
cin>>real;
cout<<"Enter the image part"<<endl;
cin>>image;
}
//member function to display the complex number.
void Complex::displayComplex(){
if (image<0) {
cout<<real<<" "<<image<<"i"<<endl;
} else {
cout<<real<<" + "<<image<<"i"<<endl;
}
}
//friend function to overload + for complex addition
Complex operator+(const Complex &A,const Complex &B){
Complex Sum;
Sum.real = A.real+B.real;
Sum.image = A.image+B.image;
return Sum;
```

```
}
//friend function to overload * for complex multiplication
Complex operator*(const Complex &A,const Complex &B){
Complex Product;
Product.real = (A.real*B.real)-(A.image*B.image);
Product.image = (A.image*B.real) + (A.real*B.image);
return Product;
}
int main() {
int quit;
cout<<"Welcome\n1.Start\n2.Quit"<<endl;
cin>>quit;

if (quit==2) {
cout<<"You have successfully quitted"<<endl;
return 0;
}
int choice,choice2;
Complex complex1,complex2,Sum,Product;
cout<<"Complex Number - 1"<<endl;
complex1.getComplex();
cout<<"Complex Number - 2"<<endl;
complex2.getComplex();
cout<<"Complex Number 1 = ";
complex1.displayComplex();
cout<<"Complex Number 2 = ";
complex2.displayComplex();
do {
cout<<"Choose the operation you want to perform\n1.Complex Addition\n2.Complex Multiplicat
cin>>choice;
switch (choice) {
case 1:
Sum = complex1+complex2;
cout<<"Sum = ";
Sum.displayComplex();
break;
case 2:
Product = complex1*complex2;
cout<<"Product = ";
Product.displayComplex();
break;
default:
break;
}
cout<<"Do you want to continue ? \n1.Continue\n2.Quit"<<endl;
cin>>choice2;
} while (choice2 == 1);
if(choice2!=1){
cout<<"Successfully Quitted!"<<endl;
}
return 0;
}
```

SAMPLE INPUT-OUTPUT



```
Welcome
1.Start
2.Quit
1
Complex Number - 1
Enter the real part
2
Enter the image part
3
Complex Number - 2
Enter the real part
4
Enter the image part
5
Complex Number 1 = 2 + 3i
Complex Number 2 = 4 + 5i
Choose the operation you want to perform
1.Complex Addition
2.Complex Multiplication
1
Sum = 6 + 8i
2
Product = -7 + 22i
```

LEAP YEAR

AIM

Write a C++ program to overload operators like *, <<, >> using friend function. The following overloaded operators should work for a class vector.

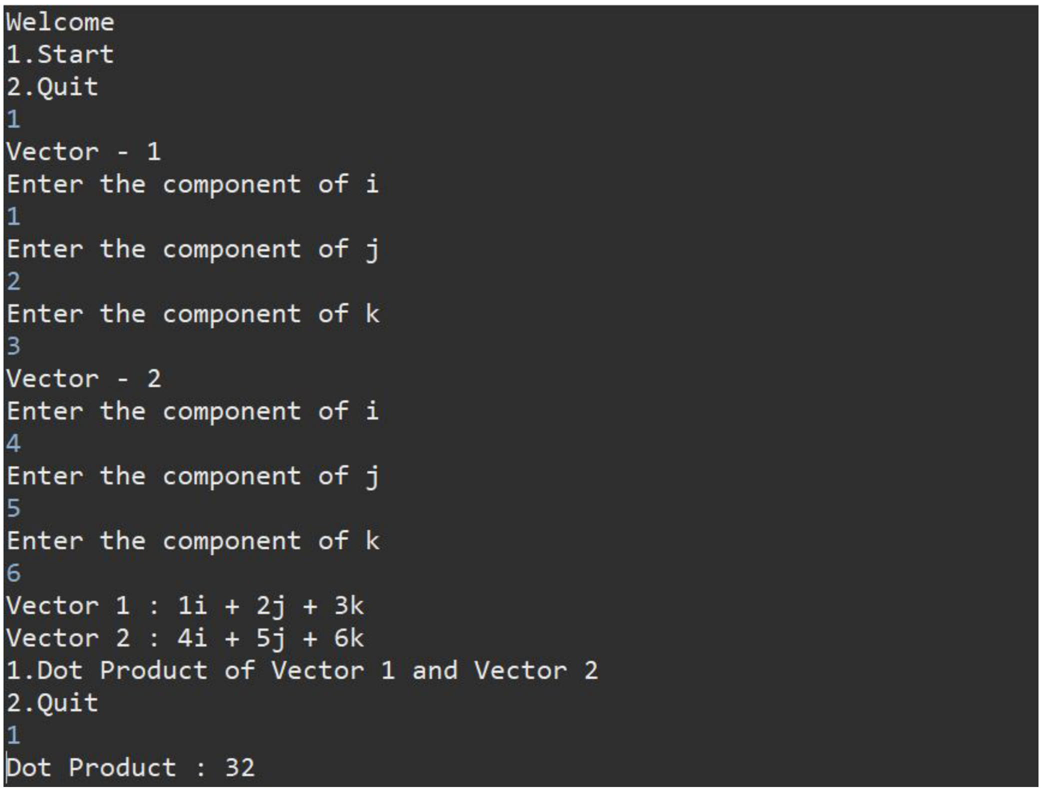
PROGRAM

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

class Vector{
float iComponent,jComponent,kComponent;
public:
float operator*(const Vector&);
friend ostream & operator<<(ostream&,Vector&);
friend istream & operator>>(istream&,Vector&);

};

//overloading >> operator to input the vector.
istream & operator>>(istream&din,Vector&a){
cout<<"Enter the component of i"<<endl;
cin>>a.iComponent;
cout<<"Enter the component of j"<<endl;
cin>>a.jComponent;
cout<<"Enter the component of k"<<endl;
cin>>a.kComponent;
return (din);
}

//overloading << operator to input the vector.
ostream & operator<<(ostream&dout,Vector&a){
if (a.jComponent>0 and a.kComponent>0) {
dout<<a.iComponent<<"i + "<<a.jComponent<<"j + "<<a.kComponent<<"k"<<endl;
}
else {
if(a.jComponent<0 and a.kComponent<0){
dout<<a.iComponent<<"i "<<a.jComponent<<"j "<<a.kComponent<<"k"<<endl;
}
else if(a.jComponent<0){
```

```
dout<<a.iComponent<<"i "<<a.jComponent<<"j + "<<a.kComponent<<"k"<<endl;
}
else {
dout<<a.iComponent<<"i + "<<a.jComponent<<"j "<<a.kComponent<<"k"<<endl;
}
}
return dout;
}

//overloading * operator to find the dot product.
float Vector::operator*(const Vector&a){
float dotProduct;
dotProduct = (iComponent*a.iComponent)+(jComponent*a.jComponent)+(kComponent*a.kComponent)
return dotProduct;
}

int main() {
int exitOption,loopOption;
cout<<"Welcome \n1.Start\n2.Quit"<<endl;
cin>>exitOption;
if (exitOption==1) {
do {
cout<<"Vector - 1"<<endl;
Vector vector1;
cin>>vector1;
cout<<"Vector - 2"<<endl;
Vector vector2;
cin>>vector2;
cout<<"Vector 1 : "<<vector1;
cout<<"Vector 2 : "<<vector2;
int choice;
cout<<"1.Dot Product of Vector 1 and Vector 2\n2.Quit"<<endl;
cin>>choice;
if(choice==1){
float dotProduct;
dotProduct = vector1*vector2;
cout<<"Dot Product : "<<dotProduct<<endl;
cout<<"Do you want to continue ?\n1.Continue\n2.Quit"<<endl;
cin>>loopOption;
}
else{
```

```
return 0;
}
} while (loopOption==1);
}
else {
return 0;
}
return 0;
}
```

SAMPLE INPUT-OUTPUT



```
Welcome
1.Start
2.Quit
1
Vector - 1
Enter the component of i
1
Enter the component of j
2
Enter the component of k
3
Vector - 2
Enter the component of i
4
Enter the component of j
5
Enter the component of k
6
Vector 1 : 1i + 2j + 3k
Vector 2 : 4i + 5j + 6k
1.Dot Product of Vector 1 and Vector 2
2.Quit
1
Dot Product : 32
```