Karnataka Law Society's
GOGTE INSTITUTE OF TECHNOLOGY
Udyambag Belagavi -590008
Karnataka, India.

Course Project Report Submission on

Solutions to Problem Statements

Submitted for the requirements of 2nd semester B.E.

for **"Object Oriented Programming using C++ (18ACS28)"**

Submitted by

	NAME	USN
1	Abhishek N Mokashi	2GI20CS003
2	Avinash V Nesargi	2GI20CS026
3	Aneesh V Kulkarni	2GI20CS011
4	Faizan Fayaz Mir	2GI20CS041

Under the guidance of

Dr. S. R. MANGALWEDE

Professor

Department of Computer Science and Engineering

Academic Year 2020-2021 (Even semester)

Karnatak Law Society's
GOGTE INSTITUTE OF TECHNOLOGY
Udyambag Belagavi -590008
Karnataka, India.

Department of Computer Science and Engineering

Certificate

This is to certify that the Course Project work titled **"Solutions to Problem Statements"**carried out by **Student** **Abhishek Mokashi**, **Avinash Nesargi, Aneesh Kulkarni, Faizan Fayaz Mir** bearing **USNs: 2GI20CS003, 2GI20CS026, 2GI20CS011, 2GI20CS041** are submittedin partial fulfilment of the requirements for 2nd semester B.E. in **COMPUTER SCIENCE AND ENGINEERING,** Visvesvaraya Technological University, Belagavi. It is certified that all corrections/suggestions indicated have been incorporated in the report. The course project report has been approved as it satisfies the academic requirements prescribed for the said degree.

Date:

Place: Belagavi

Signature of Guide

Dr. S. R. Mangalwede

Professor

KLS Gogte Institute of Technology, Belagavi

Name of the Examiners

1._____

Signature of the Examiners

1._____

Table of contents

1. Consider n students' records, each consisting of name, USN and marks in three tests and average. Develop a C++ program to read name, USN and marks in three subjects. Write a function that updates the average field (using reference parameter) by computing average of best two marks. Display the updated student records.

SOURCE CODE:-

```
struct Student {
    string name, USN;
    int m1, m2, m3;
    int avg;
};
void computeAvg (int m1, int m2, int m3, int &avg) {
    int smallest = m1<m2 ? (m1<m3 ? m1 : m3) : m2<m3 ? m2 : m3;
    avg = ceil((m1+m2+m3-smallest) / 2.0);
}
void readData(Student st[], int n) {
    int i;
    cout<< "Enter data for " << n << " students:" <<endl;
    for(i=0; i<n; i++) {
            cout<< "Name:";
            cin>>st[i].name;
            cout<< "USN:";
            cin>>st[i].USN;
            fflush(stdin);
```

```
            cout<< "Enter three marks:";
            cin>>st[i].m1 >>st[i].m2 >>st[i].m3;
        }
    }
    void printDetails(Student st[], int n) {
        cout<< "Student details:" <<endl;
        for(int i=0; i<n; i++) {
        cout<<st[i].name << " " <<st[i].USN<< " " <<st[i].avg <<endl;
        }
    }
    int main() {
        int n;
        cout<< "How many students?";
        cin>> n;
        Student st[n];
        readData(st, n);
        for(int i=0; i<n;i++)
            computeAvg(st[i].m1, st[i].m2, st[i].m3, st[i].avg);
        printDetails(st, n);
        return 0;
    }
```

OUTPUT:-

```
How many students?3
Enter data for 3 students:
Name:Aneesh
USN:2GI20CS011
Enter three marks:45 34 43
Name:Avinash
USN:2GI20CS026
Enter three marks:50 42 47
Name:Abhishek
USN:2GI20CS003
Enter three marks:45 38 48
Student details:
Aneesh 2GI20CS011 44
Avinash 2GI20CS026 48
Abhishek 2GI20CS003 46


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

2. Write a C++ program to calculate maturity value of investment using function overloading. One function computes the maturity value using simple interest formula and the other function computes the maturity value using compound interest formula.

   Formula for maturity value using simple interest calculation: SI = p + (p * t * r / 100)

   Formula for maturity value using compound interest: $A = P (1 + R/(100 * n))^{nt}$

   where,

   p is the principal amount

   r is the rate of interest

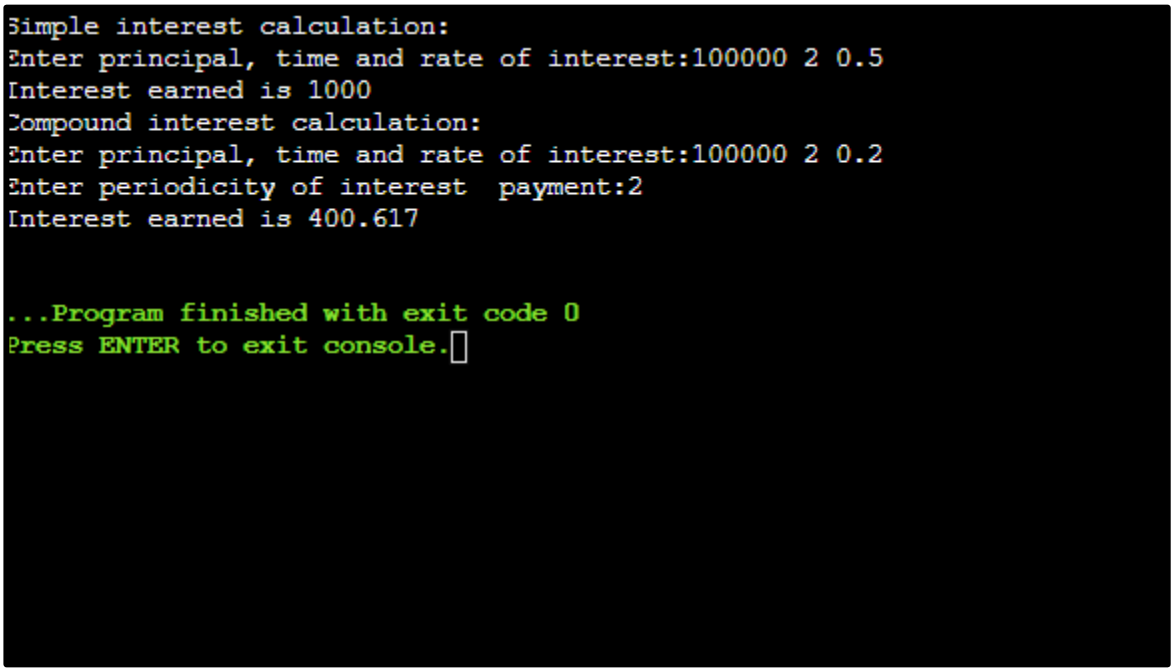
   t is the time duration of investment

   n is the number of times interest is compounded per year

SOURCE CODE:-

```
float computeInterest(int p, int t, float r) {
    return p*t*r/100;
}
float computeInterest(int p, int t, float r, int n) {
    float findAmount;
    findAmount = p * pow((1 + r /(100 * n)),(n*t));
    return findAmount - p;
}
int main() {
    int p, t, n;
    float r;
    cout<< "Simple interest calculation:" <<endl;
    cout<< "Enter principal, time and rate of interest:";
    cin>> p >> t >> r;
    cout<< "Interest earned is " <<computeInterest(p,t,r) <<endl;
    cout<< "Compound interest calculation:" <<endl;
    cout<< "Enter principal, time and rate of interest:";
    cin>> p >> t >> r;
    cout<< "Enter periodicity of interest  payment:";
    cin>> n;
    cout<< "Interest earned is " <<computeInterest(p,t,r,n) <<endl;
    return 0;
}
```

OUTPUT:-


Simple interest calculation:
Enter principal, time and rate of interest:100000 2 0.5
Interest earned is 1000
Compound interest calculation:
Enter principal, time and rate of interest:100000 2 0.2
Enter periodicity of interest payment:2
Interest earned is 400.617
...Program finished with exit code 0
Press ENTER to exit console.

3. Create a class to represent a bank account. Include the following members:

   Data members:

   a. name of the depositor

   b. account number

   c. balance amount

   Member functions:

   a. To assign initial values

   b. To deposit an amount

   c. To withdraw an amount after checking the balance

   d. To display details of the account

Write a menu driven main function that calls the member functions until the user chooses to terminate the program.

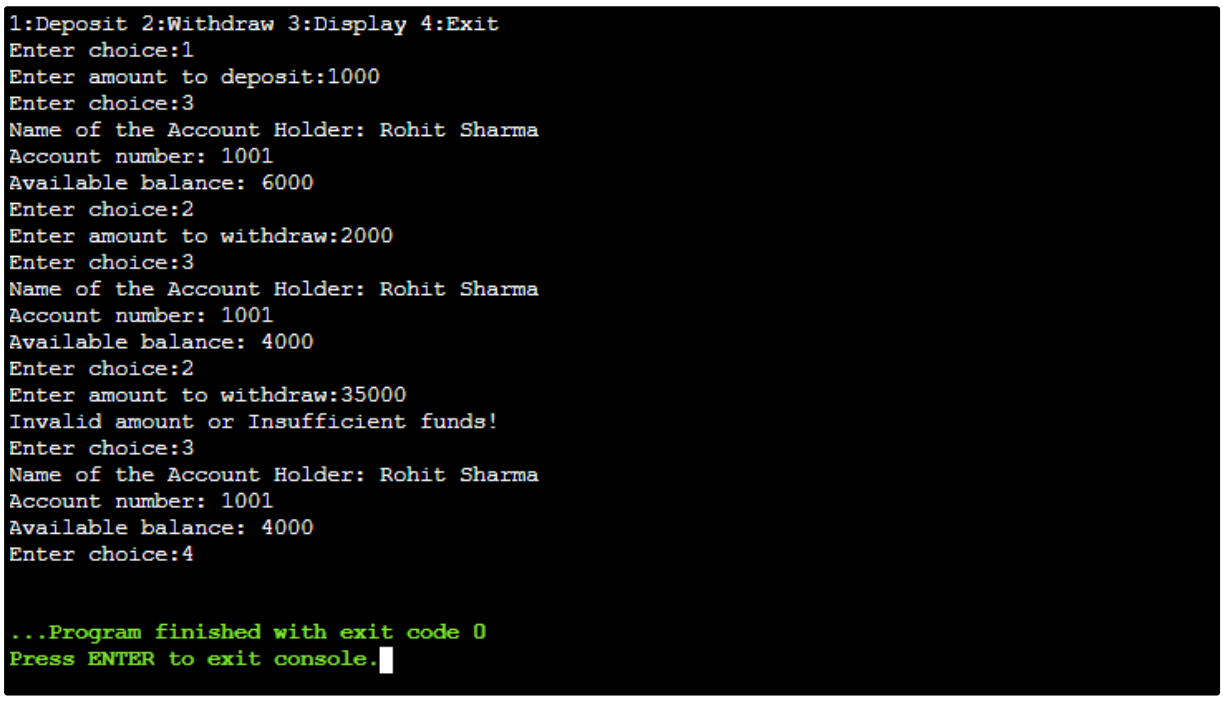
SOURCE CODE:-

```
class Account {
private:
    string name;
    int accNo, balance;
public:
    void initalize(string name, int num, int bal) {
        this->name = name;
        accNo = num;
        balance = bal;
    }
    void deposit(int amt) {
        if(amt>=0)
            balance += amt;
        else
            cout<< "Invalid operation! Amount cannot be negative"
<<endl;
    }
    void withdraw(int amt) {
        if(amt>=0 && balance-amt>=1000)
            balance -= amt;
        else
            cout<<"Invalid amount or Insufficient funds!" <<endl;
    }
```

```
    void display() {
        cout<< "Name of the Account Holder: " << name <<endl;
        cout<< "Account number: " <<accNo<<endl;
        cout<< "Available balance: " << balance <<endl;
    }
};
int main() {
    Account ac;
    ac.initalize("Rohit Sharma", 1001, 5000);
    int choice, amt;
    cout<< "1:Deposit 2:Withdraw 3:Display 4:Exit" <<endl;
    do {
        cout<< "Enter choice:";
        cin>> choice;
        switch(choice) {
            case 1:
                cout<< "Enter amount to deposit:";
                cin>> amt;
                ac.deposit(amt);
                break;
            case 2:
                cout<< "Enter amount to withdraw:";
                cin>> amt;
                ac.withdraw(amt);
```

```
                    break;
                case 3:
                    ac.display();
                    break;
                case 4:
                    break;
                default:
                    cout<< "Invalid choice! Try again!!" <<endl;
            }
    }while(choice!=4);
    return 0;
}
```

OUTPUT:-



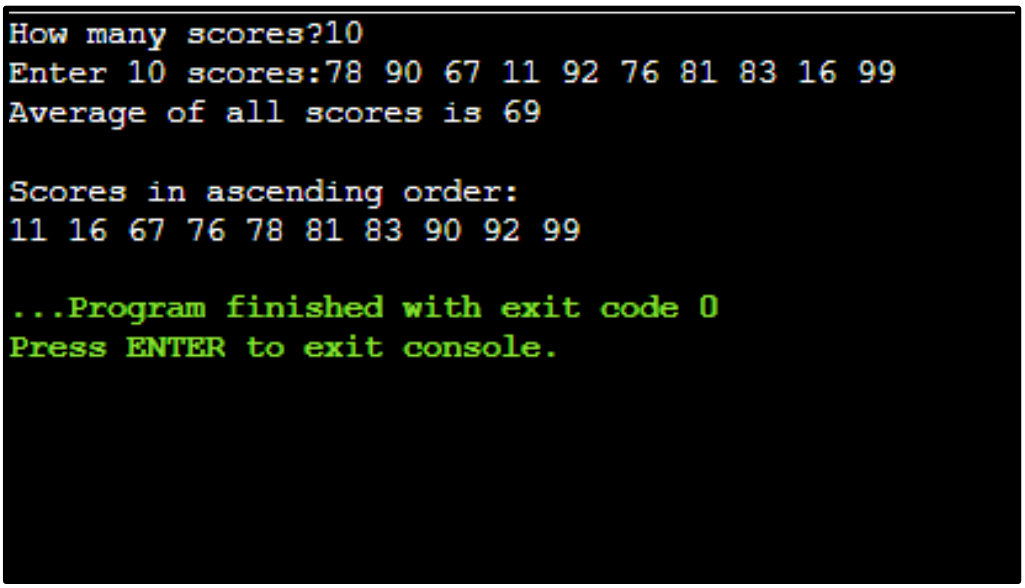
4. Create a C++ application that receives the test scores from the user. The application computes the average of test scores. It also displays the scores in ascending order. The application does dynamic allocation of memory for storing the scores.

SOURCE CODE:-

```
void readScores(int a[], int n) {
    cout<< "Enter " << n << " scores:";
    for(int i=0; i<n; i++)
        cin>> a[i];
}
void findAverage(int a[], int n) {
    int sum = 0;
    for(int i=0; i<n; i++)
        sum += a[i];
    cout<< "Average of all scores is " << sum/n <<endl;
}
void sortAnddisplay(int a[], int n) {
    for(int i=0; i<n-1; i++)
        for(int j=0; j<n-i-1; j++)
            if(a[j]>a[j+1]) {
                int temp = a[j];
                a[j] = a[j+1];
                a[j+1] = temp;
            }
```

```
    cout<<endl<< "Scores in ascending order:" <<endl;
    for(int i=0; i<n; i++)
            cout << a[i]<<” ”;
}
int main() {
    int n;
    cout<< "How many scores?";
    cin>> n;
    int *a = new int[n];
    readScores(a, n);
    findAverage(a, n);
    sortAnddisplay(a, n);
    return 0;
}
```

OUTPUT:-



5. Develop a C++ program consisting of an array of n Movie objects with each Movie having name, year of release and rating as data members. Include member functions to:
    a. set values for data members and
    b. search for a particular movie based on name
    c. display movie details

SOURCE CODE:-

```
class Movie {
private:
    string name;
    int year;
    float rating;
public:
    void getMovieData() {
        fflush(stdin);
        cout<<endl<< "Enter movie name:";
        getline(cin, name);
        fflush(stdin);
        cout<< "Year of release:";
        cin>> year;
        cout<< "Rating:";
        cin>> rating;
    }
    void dispMovieDetails() {
```

```
            cout<< "Name:" << name <<endl;
            cout<< "Year of  release:" << year <<endl;
            cout<< "Rating:" << rating <<endl;
        }
        friend void searchMovie(Movie m[], int n, string name);
        friend void topMovie(Movie  m[], int n);
};
void searchMovie(Movie m[], int n, string name) {
        int found = 0, i;
        for(i=0; i<n; i++)
            if(m[i].name.compare(name)==0) {
                found = 1;
                break;
            }
        if(found==0)
            cout<< "Such movie does not exist!" <<endl;
        else {
            cout<< "Movie found. Details are..." <<endl;
            cout<< "Year of release:" << m[i].year<<endl;
            cout<< "Rating:" << m[i].rating<<endl;
        }
}
void topMovie(Movie m[], int n) {
        float top = m[0].rating;
```

```
	int movNum;
	for(int i=1; i<n; i++)
		if(m[i].rating> top) {
			top = m[i].rating;
			movNum = i;
		}
	cout<< "Movie with top rating is " << m[movNum].name <<endl;
	cout<< "Year of release:" << m[movNum].year<<endl;
	cout<< "Rating:" << top <<endl;
}
int main() {
	int n;
	cout<< "How many movies? ";
	cin>> n;
	fflush(stdin);
	Movie *m = new Movie[n];
	cout<< "Enter details of " << n << " movies:";
	for(int i=0; i<n; i++)
		m[i].getMovieData();
	cout<< "Movie data..." <<endl;
	for(int i=0; i<n; i++)
		m[i].dispMovieDetails();
	string name;
	fflush(stdin);
```

```
        cout<< "Enter movie name to search:";
        getline(cin, name);
        searchMovie(m, n, name);
        topMovie(m, n);
        return 0;
}
```

OUTPUT:-

```
PS C:\Users\FAIZAN FAYAZ MIR\Desktop\cpp> cd "c:\Use
sem } ; if ($?) { .\termwork2ndsem }
How many movies? 2
Enter details of 2 movies:
Enter movie name:Inception
Year of release:2010
Rating:9

Enter movie name:Interseller
Year of release:2016
Rating:8.9
Movie data...
Name:Inception
Year of  release:2010
Rating:9
Name:Interseller
Year of  release:2016
Rating:8.9
Enter movie name to search:Inception
Movie found. Details are...
Year of release:2010
Rating:9
Movie with top rating is Inception
Year of release:2010
Rating:9
PS C:\Users\FAIZAN FAYAZ MIR\Desktop\cpp>
```

6. Create a class called intArray with

   Data members:

   1. pointer to an integer array and
   2. integer to hold the array length

   Member functions:

   1. A zero-arg constructor
   2. A parameterized constructor with an array and its length as parameters
   3. A copy constructor
   4. Display array elements
   5. Destructor

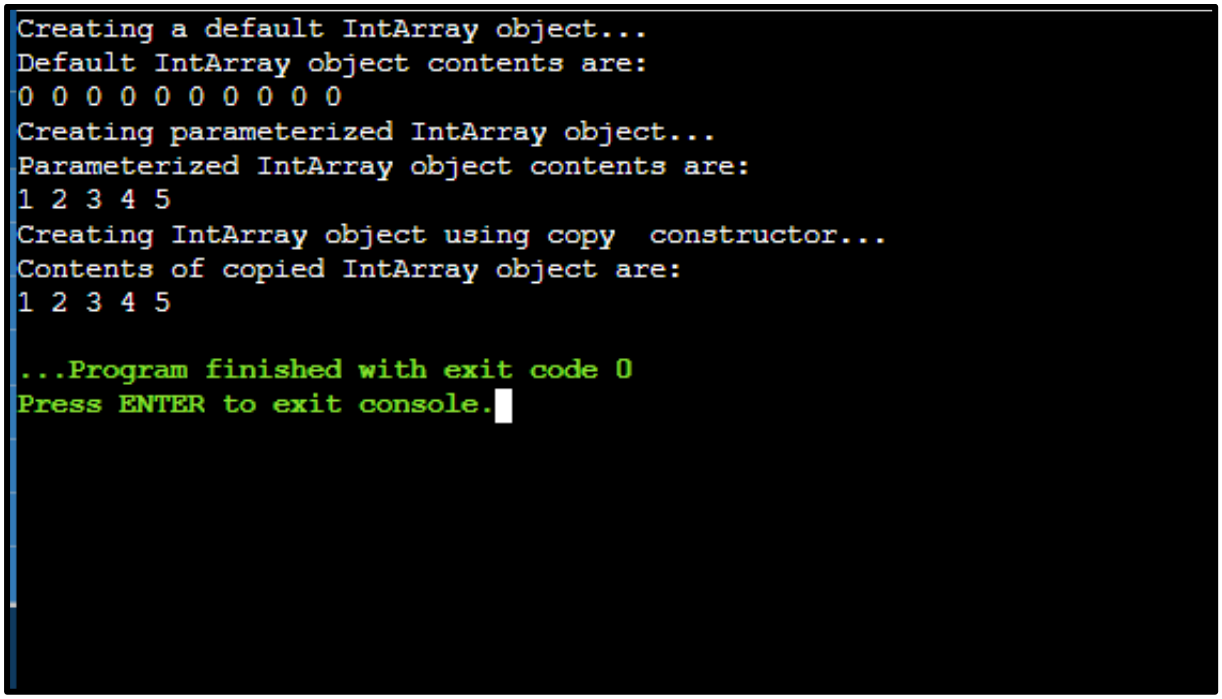
   Write the corresponding main()

SOURCE CODE:-

```
class IntArray {
private:
    int len, *arr;
public:
    IntArray() {
            len = 10;
            arr = new int[len];
            for(int i=0; i<len; i++)
                    arr[i]=0;
    }
    IntArray(int a[], int n) {
            len = n;
            arr = new int[len];
            for(int i=0; i<len; i++)
                    arr[i] = a[i];
    }
    IntArray(const IntArray&a) {
            len = a.len;
            arr = new int[len];
            for(int i=0; i<len; i++)
                    arr[i] = a.arr[i];
    }
    void printArray() {
```

```
            for(int i=0; i<len; i++)
                cout<<setw(3) <<arr[i];
    }
};
int main() {
    cout<< "Creating a default IntArray object..." <<endl;
    IntArray ob1;
    cout<< "Default IntArray object contents are:" <<endl;
    ob1.printArray();
    cout<<endl<< "Creating parameterized IntArray object..." <<endl;
    int a[] = {1,2,3,4,5};
    IntArray ob2(a,5);
    cout<< "Parameterized IntArray object contents are:" <<endl;
    ob2.printArray();
    cout<<endl<< "Creating IntArray object using copy  constructor..."
<<endl;
    IntArray ob3(ob2);
    cout<< "Contents of copied IntArray object are:" <<endl;
    ob3.printArray();
    return 0;
}
```

OUTPUT:-


Creating a default IntArray object...
Default IntArray object contents are:
0 0 0 0 0 0 0 0 0 0
Creating parameterized IntArray object...
Parameterized IntArray object contents are:
1 2 3 4 5
Creating IntArray object using copy constructor...
Contents of copied IntArray object are:
1 2 3 4 5
...Program finished with exit code 0
Press ENTER to exit console.

7. Create a class to represent a shape named Box. Include the following members length, breadth and height.

   It includes the member functions to set length, breadth and height. Compute the volume of two boxes.

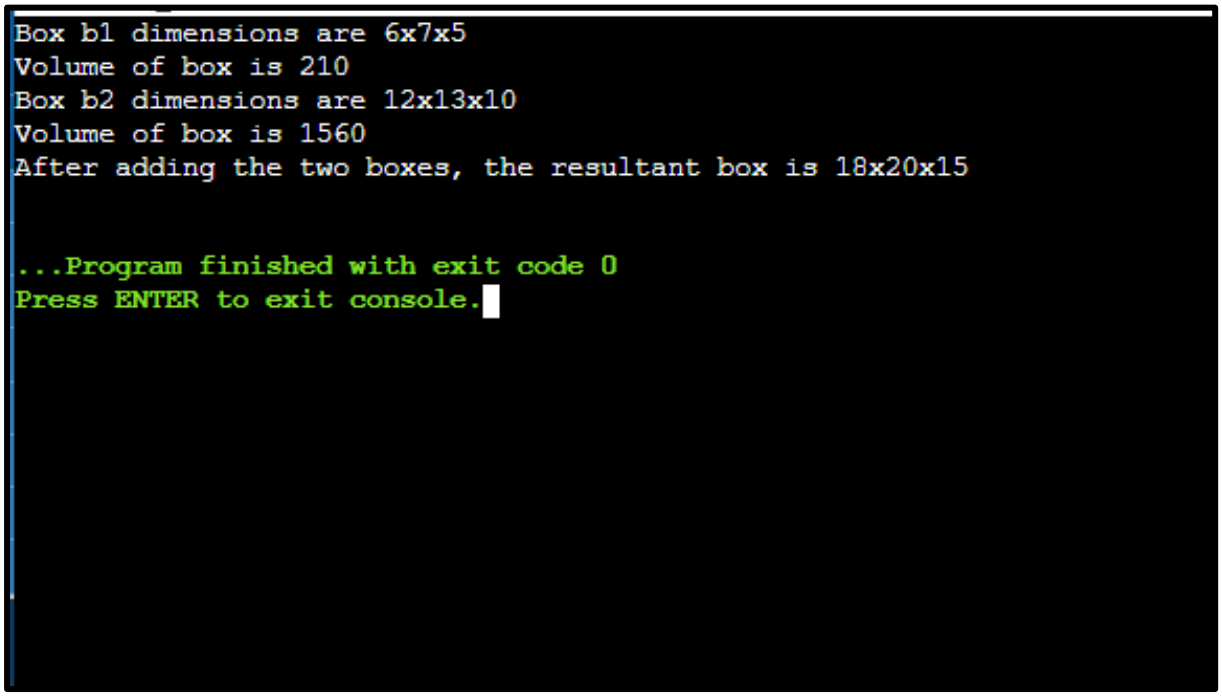
   Overload (+) operator to add two Box objects and store the result in Box3.

SOURCE CODE:-

```
class Box {
private:
    int len, bdth, hgt;
public:
    void setDimensions(int l, int b, int h) {
            len = l;
            bdth = b;
            hgt = h;
    }
    void computeVolume() {
            cout<< "Volume of box is " <<len * bdth * hgt<<endl;
    }
    void display() {
            cout<<len<< "x" <<bdth<< "x" <<hgt<<endl;
    }
    friend Box operator+(Box b1, Box b2);
```

```
};
Box operator +(Box b1, Box b2) {
    Box res;
    res.len = b1.len + b2.len;
    res.bdth = b1.bdth + b2.bdth;
    res.hgt = b1.hgt + b2.hgt;
    return res;
}
int main() {
    Box b1, b2;
    b1.setDimensions(6,7,5);
    cout<< "Box b1 dimensions are ";
    b1.display();
    b1.computeVolume();
    b2.setDimensions(12,13,10);
    cout<< "Box b2 dimensions are ";
    b2.display();
    b2.computeVolume();
    Box b3;
    b3 = b1 + b2;
    cout<< "After adding the two boxes, the resultant box is ";
    b3.display();
    return 0;
}
```

OUTPUT:-


Box b1 dimensions are 6x7x5
Volume of box is 210
Box b2 dimensions are 12x13x10
Volume of box is 1560
After adding the two boxes, the resultant box is 18x20x15
...Program finished with exit code 0
Press ENTER to exit console.

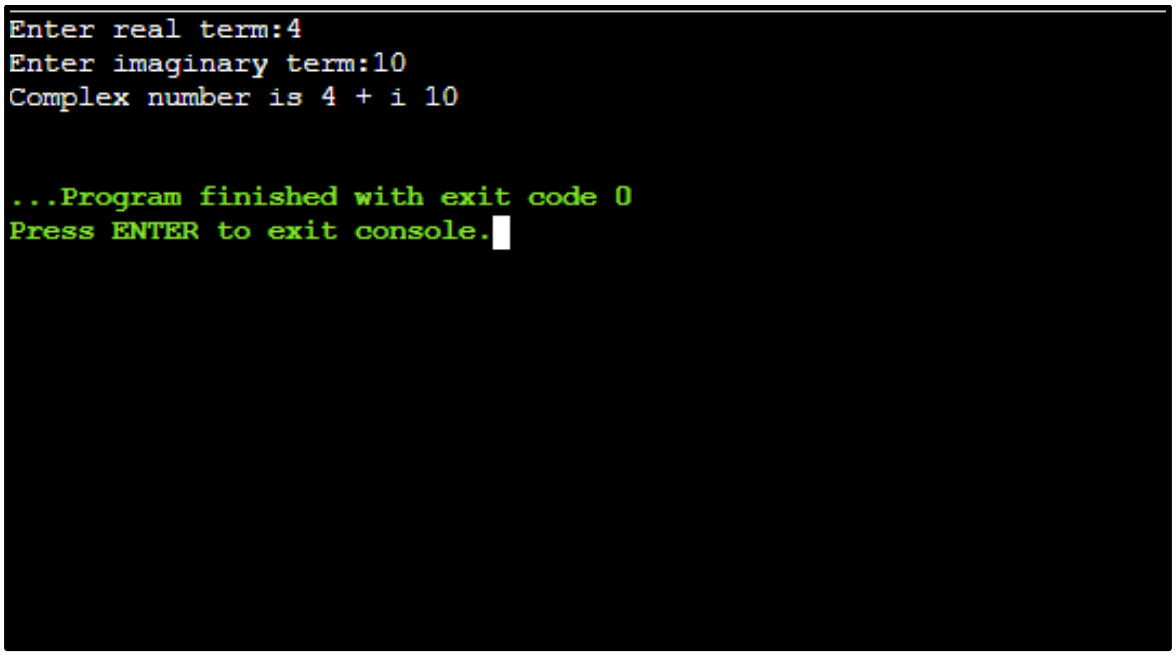
8. Create a class called complex. It has data members as real and imag. It reads real and imag values by overloading << operator. It also displays the value of the complex object by overloading >> operator.

SOURCE CODE:-

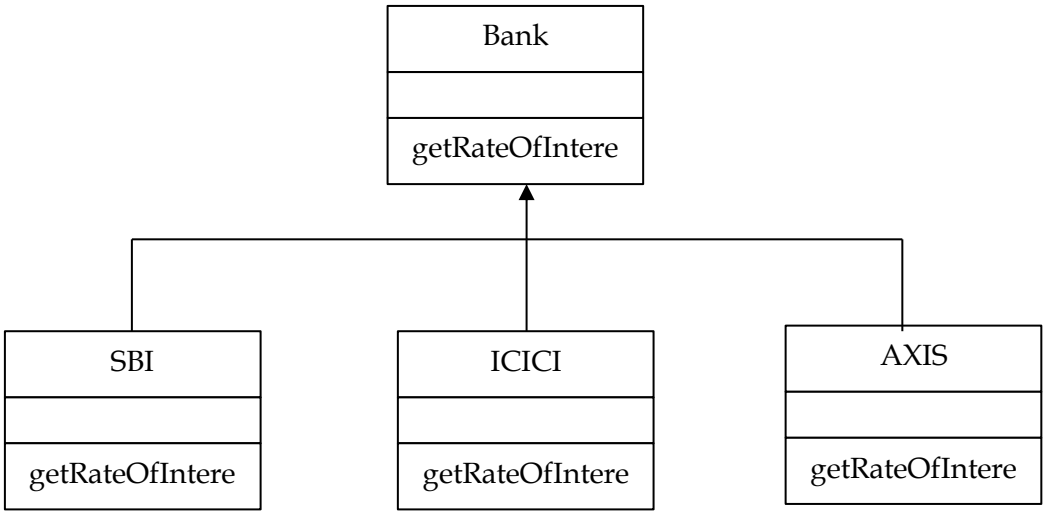
```
class Complex {
private:
    int real, imag;
public:
    friend ostream&operator<<(ostream&out, const Complex &c) {
            out<< "Complex number is ";
            out<<c.real<< " + i " <<c.imag<<endl;
            return out;
    }
    friend istream&operator>>(istream&in, Complex &c) {
            cout<< "Enter real term:";
            in>>c.real;
            cout<< "Enter imaginary term:";
            in>>c.imag;
            return in;
    }
};
int main() {
    Complex c1;
```

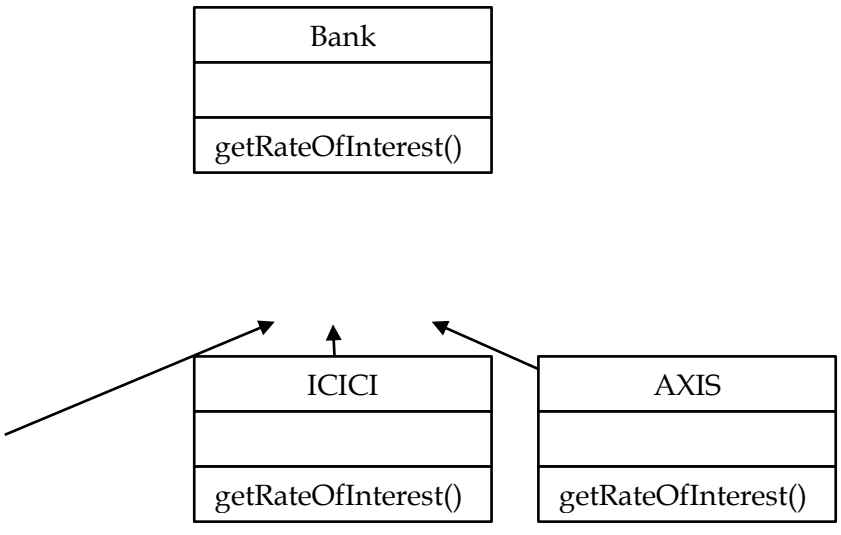
```
    cin>> c1;
    cout<< c1;
    return 0;
}
```

OUTPUT:-



9. Implement a BANK class to demonstrate the inheritance in C++ by implementing getRateOfInterest member function for three different banks, as shown below.



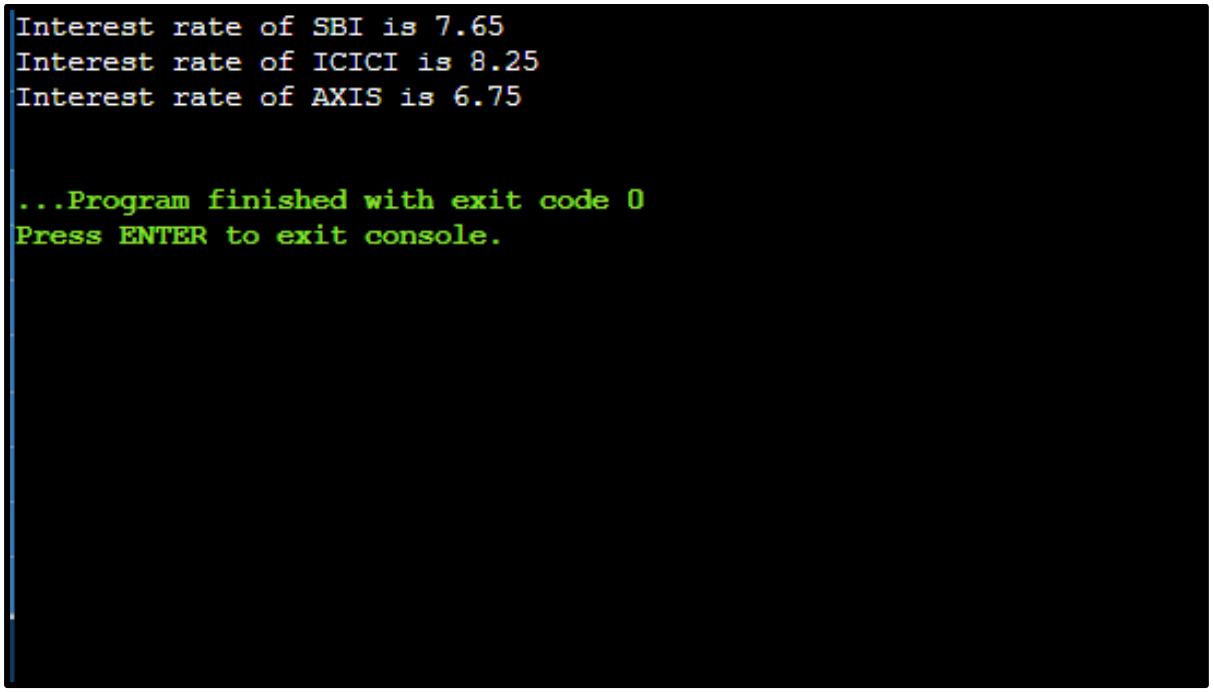


SOURCE CODE:-

```
class Bank {
protected:
    float intRate;
public:
    Bank(float intRate) {
        this->intRate = intRate;
    }
    float getRateOfInterest() {
        cout << "This function should be overridden by subclasses!"
<< endl;
    }
};
class SBI : public Bank {
public:
    SBI(float intRate) : Bank(intRate) {}
    float getRateOfInterest() {
        return intRate;
    }
};
class ICICI : public Bank {
public:
    ICICI(float intRate) : Bank(intRate) {}
    float getRateOfInterest() {
```

```
        return intRate;
    }
};
class AXIS : public Bank {
public:
    AXIS(float intRate) : Bank(intRate) {}
    float getRateOfInterest() {
        return intRate;
    }
};
int main() {
    SBI ac1(7.65);
    ICICI ac2(8.25);
    AXIS ac3(6.75);
    cout << "Interest rate of SBI is " << ac1.getRateOfInterest() << endl;
    cout << "Interest rate of ICICI is " << ac2.getRateOfInterest() <<
endl;
    cout << "Interest rate of AXIS is " << ac3.getRateOfInterest() <<
endl;
    return 0;
}
```

OUTPUT:-


Interest rate of SBI is 7.65
Interest rate of ICICI is 8.25
Interest rate of AXIS is 6.75
...Program finished with exit code 0
Press ENTER to exit console.

10. Create three classes *Person, Professor* and *Student*. The class *Person* should have data members name and age. The classes *Professor* and *Student* should inherit from the class *Person*.

The class *Professor* should have two integer members: *publications* and cur_id. There will be two member functions: *getdata* and *putdata*. The function *getdata* should get the input from the user: the *name, age* and *publications* of the professor. The function *putdata* should print the *name, age, publications* and the cur_id of the professor.

The class *Student* should have two data members: *marks*, which is an array of 3 integers and cur_id. It has two member functions: *getdata* and *putdata*. The function *getdata* should get the input from the user: the *name, age*, and the *marks* of the student in 3 subjects. The function *putdata* should print the *name, age*, *sum* of the marks and the cur_id of the student.

For each object being created of the *Professor* or the *Student* class, sequential id's should be assigned to them starting from 1.

Solve this problem using virtual functions, constructors and static variables.

SOURCE CODE:-

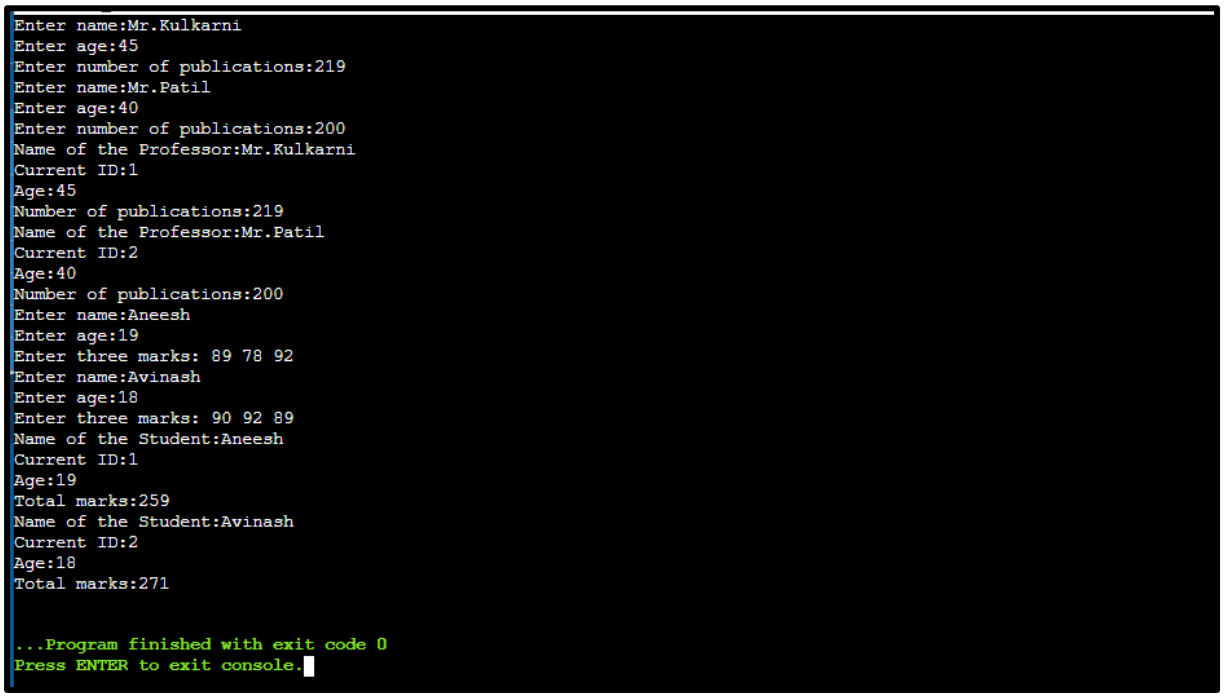
```
class Person {
protected:
	string name;
	int age, prof_id, stud_id;
public:
	virtual void putdata() = 0;
	virtual void getdata() = 0;
};
class Professor : public Person {
private:
	int publications;
	static int cur_id;
public:
	Professor() {
		prof_id = cur_id++;
	}
	void getdata() {
		cout << "Enter name:";
		cin >> name;
		cout << "Enter age:";
		cin >> age;
		cout << "Enter number of publications:";
		cin >> publications;
```

```
        }
        void putdata() {
            cout << "Name of the Professor:" << name << endl;
            cout << "Current ID:" << prof_id << endl;
            cout << "Age:" << age << endl;
            cout << "Number of publications:" << publications <<
endl;
        }
    };
    int Professor :: cur_id = 1;
    class Student : public Person {
    private:
        int marks[3];
        static int cur_id;
    public:
        Student() {
            stud_id = cur_id++;
        }
        void getdata() {
            cout << "Enter name:";
            cin >> name;
            cout << "Enter age:";
            cin >> age;
            cout << "Enter three marks: ";
```

```
            for(int i=0; i<3; i++)
                cin >> marks[i];
        }
        void putdata() {
            cout << "Name of the Student:" << name << endl;
            cout << "Current ID:" << stud_id << endl;
            cout << "Age:" << age << endl;
            int sum = 0;
            for(int i=0; i<3; i++)
                sum += marks[i];
            cout << "Total marks:" << sum << endl;
        }
};
int Student :: cur_id = 1;
int main() {
        Professor p1, p2;
        p1.getdata();
        p2.getdata();
        p1.putdata();
        p2.putdata();
        Student s1, s2;
        s1.getdata();
        s2.getdata();
        s1.putdata();
```

```
        s2.putdata();
        return 0;
    }
```

OUTPUT:-



11. Create a base class called Polygon and three derived classes: Triangle, Square and Rectangle. Demonstrate dynamic polymorphism to compute the area of a triangle, square and rectangle.

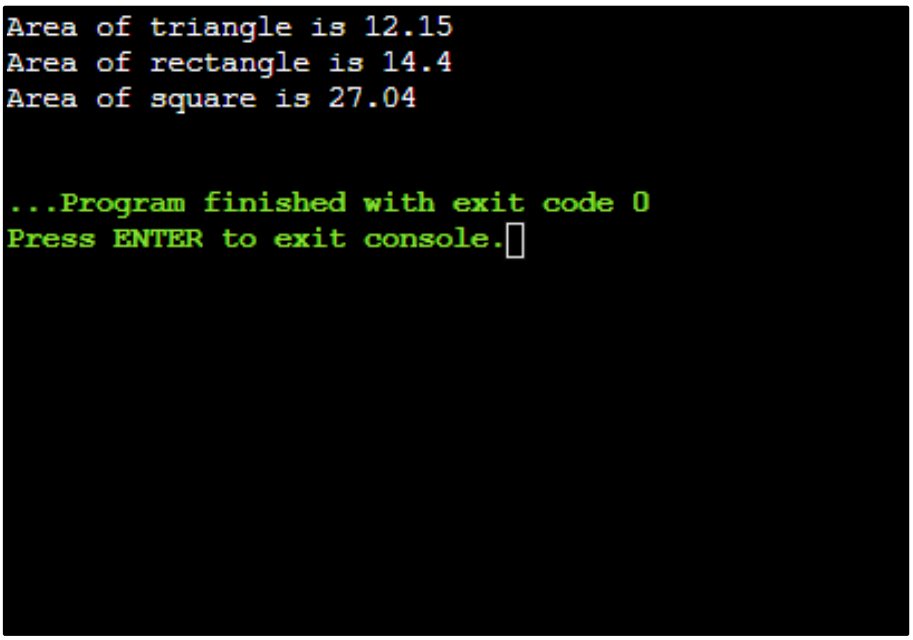
SOURCE CODE:-

```
class Polygon {
protected:
    float area;
public:
    virtual float computeArea() = 0;
};
class Triangle : public Polygon {
private:
    float base, height;
public:
    Triangle(float base, float height) {
        this->base = base;
        this->height = height;
    }
    float computeArea() {
        area = 0.5 * base * height;
        return area;
    }
};
```

```
class Rectangle : public Polygon {
private:
	float length, width;
public:
	Rectangle(float length, float width) {
		this->length = length;
		this->width = width;
	}
	float computeArea() {
		area = length * width;
		return area;
	}
};
class Square : public Polygon {
private:
	float side;
public:
	Square(float side) {
		this->side = side;
	}
	float computeArea() {
		area = side * side;
		return area;
	}
```

```
};
int main() {
        Polygon *p;
        Triangle t1(4.5, 5.4);
        Rectangle r1(4.0, 3.6);
        Square s1(5.2);
        p = &t1;
        cout << "Area of triangle is " << p->computeArea() << endl;
        p = &r1;
        cout << "Area of rectangle is " << p->computeArea() << endl;
        p = &s1;
        cout << "Area of square is " << p->computeArea() << endl;
        return 0;
}
```

OUTPUT:-



12. Write a C++ program to copy the contents of two files (first followed by second) into a third file.

SOURCE CODE:-

```
int main() {
    char ch;
    int a=0, d=0, sp=0;
    ifstream inFile1("file1.txt");
    ifstream inFile2("file2.txt");
    ofstream outFile("final.txt");
    while(inFile1.get(ch)) {
            outFile.put(ch);
    }
    while(inFile2.get(ch)) {
            outFile.put(ch);
    }
    inFile1.close();
    inFile2.close();
    outFile.close();
    return 0;
}
```

OUTPUT:-


PS C:\Users\FAIZAN FAYAZ MIR\Desktop\cpp> cd "c:\Users\FAIZAN FAYAZ MIR
sem } ; if ($?) { .\termwork2ndsem }
PS C:\Users\FAIZAN FAYAZ MIR\Desktop\cpp>