1. Single Inheritance

A class Room consists of two fields length and breadth and method int area() to find the area of room. A new class BedRoom is derived from class Room and consist of additional field height and two methods setData (int,int,int) to set the value for three fields and int volume() to find the volume.

Now WAP to input the length, breadth and height and find the area and volume.

```
#include<iostream>
using namespace std;
class Room
{
        protected:
                float length, breadth;
        public:
                int area()
{
return(length*breadth);
}
};
class BedRoom : public Room
{
        private:
                float height;
        public:
        void setData(int l, int b, int h)
        {
length=l;
breadth=b;
```

```
height=h;
        }
int volume()
{
return(length * breadth * height);
}
};

int main()
{
BedRoom b;
b.setData(3,4,5);
cout<<"Area of bedroom= "<<b.area()<<endl;
cout<<"Volume of bedroom="<<b.volume();
}
```

---

2. Multiple Inheritance

Create two classes class1 and class2 each having data member for storing a number, a method to initialize it. Create a new class class3 that is derived from both class class1 and class2 and consisting of a method that displays the sum of two numbers from class1 and class2.

```
#include<iostream>
using namespace std;
class class1
{
protected:
        int n;
public:
```

```
        void getn(int p)
{
n=p;
}
};


class class2
{
protected:
        int m;
public:
        void getm(int q)
        {
        m=q;
        }
};
class class3: public class1, public class2
{
public:
        void displaytotal()
        {
        int tot;
        tot=n+m;
        cout<<"Sum ="<<tot;
        }
};
```

```
int main()
{
class3 a;
a.getm(4);
a.getn(5);
a.displaytotal();
}
```

---

3. Multilevel inheritance

A class Student consists of field roll, a method to assigns roll number. A new class Test is derived from class Student and consists of two new fields sub1 and sub2, a method to initialize these fields with obtained mark. Again, a new class Result is derived from Test and consists of a field total and a method to display entire details along with total obtained marks. WAP to input roll number, marks in two different subject and display total.

```
#include<iostream>
using namespace std;
class Student
{
protected:
        int roll;
public:
        void setroll(int r)
        {
        roll=r;
        }
};
```

```
class Test: public Student
{
protected:
        float sub1, sub2;
public:
void setmark(float m1, float m2)
{
        sub1=m1;
        sub2=m2;
}
};
class Result : public Test
{
private:
        float total;
public:
void display()
{
total=sub1+sub2;
cout<<"Roll number= "<<roll;
cout<<"Mark in first subject= "<<sub1;
cout<<"Mark in second subject= "<<sub2;
cout<<"Total= "<<total;
}
};

int main()
```

```
{
int r;
float s1,s2;
cout<<"Enter roll number";
cin>>r;
cout<<"Enter marks in two subject";
cin>>s1>>s2;
Result res;
res.setroll(r);
res.setmark(s1,s2);
res.display();
}
```

---

4. Hierarchical inheritance

A company needs to keep record of its following employees:

i) Manager ii) Supervisor

The record requires name and salary of both employees. In addition, it also requires section_name (i.e. name of section, example Accounts, Marketing, etc.) for the Manager and group_id (Group identification number, e.g. 205, 112, etc.) for the Supervisor. Design classes for the above requirement. Each of the classes should have a function called set() to assign data to the fields and a function called get() to return the value of the fields. Write a main program to test your classes.

```
#include<iostream>
#include<string.h>
using namespace std;
class Employee
{
private:
```

```
        char name[30];
        float salary;
public:
void setName(char *n)
{
        strcpy(name,n);
}
void setSalary(float s)
{
        salary=s;
}
char *getName()
{
        return name;
}
float getSalary()
{
        return salary;
}
};
class Manager: public Employee
{
private:
        char section_name[50];
public:
void setSection_name(char *sn)
{
```

```
strcpy(section_name,sn);
}
char * getSection_name()
{

return section_name;
}
};
class Supervisor: public Employee
{
private:
        int group_id;
public:
void setGroup_id(int gid)
{
        group_id=gid;
}
int getGroup_id()
{
        return group_id;
}
};

int main()
{
Manager m;
m.setName("Elon Musk");
```

```
m.setSalary(50000);
m.setSection_name("Accounts");
cout<<"Name= "<<m.getName();
cout<<"Salary= "<<m.getSalary();
cout<<"Section= "<<m.getSection_name();
Supervisor s;
s.setName("Bill Gates");
s.setSalary(40000);
s.setGroup_id(5);
cout<<"Name= "<<s.getName();
cout<<"Salary= "<<s.getSalary();
cout<<"Group ID= "<<s.getGroup_id();
}
```

---

5. Hybrid inheritance

Create a class Student with data member roll_no and two functions to initialize and display it. Derive a new class Test which has two methods to assign and display marks in two subjects. Create a new class Sport with two functions that assign and display the score in sports. Now create another class Result that is derived from both class Test and Sport, having a function that displays the total of marks and score. Write a main program to test your class.

The example shows the hybrid inheritance i.e. combination of multilevel and multiple inheritance.

```
#include<iostream>
using namespace std;
class Student
{
private:
int roll;
```

```
public:
void setroll()
{
cout<<"Enter roll number";
cin>>roll;
}
void showroll()
{
cout<<"Roll= "<<roll;
}
};
class Test: public Student
{
protected:
float com,eng;
public:
void setmark()
{
cout<<"Enter marks of computer and English ";
cin>>com>>eng;
}
void showmark()
{
cout<<"Computer= "<<com;
cout<<"English= "<<eng;
}
};
```

```
class Sport
{
protected:
float score;
public:
void setscore()
{
cout<<"Enter score in sports ";
cin>>score;
}
void showscore()
{
cout<<"Score in sports= "<<score;
}
};
class Result: public Test, public Sport
{
private:
float tot;
public:
void showtotal()
{
tot=com+eng+score;
cout<<"Total obtained marks= "<<tot;
}
};
```

```
int main()
{
Result res;
res.setroll();
res.setmark();
res.setscore();
res.showroll();
res.showmark();
res.showscore();
res.showtotal();
}
```

Define a shape class (with necessary constructors and member functions) in Object Oriented Programming (abstract necessary attributes and their types). Write a complete code in C++ programming language.

i. Derive triangle and rectangle classes from shape class adding necessary attributes.

ii. Use these classes in main function and display the area of triangle and rectangle.

```
#include<iostream>
using namespace std;
class shape
{
protected:
float breadth, height, area;
public:
void getshapedata()
{
cout<<"Enter breadth:";
cin>>breadth;
```

```
cout<<"Enter height:";
cin>>height;
}
};
class triangle: public shape
{
public:
void calarea()
{
area=(breadth * height)/2;
}
void display()
{
cout<<"The area of triangle is"<<area;
}
};
class rectangle: public shape
{
public:
void calarea()
{
area=breadth * height;
}
void display()
{
cout<<"Area of rectangle is"<<area;
}
```

```
};
int main()
{
triangle T;
rectangle R;
cout<<"Enter triangle data:";
T.getshapedata();
cout<<"Enter rectangle data:";
R.getshapedata();
T.calarea();
R.calarea();
T.display();
R.display();
}
```

Define a student class (with necessary constructors and member functions) in Object Oriented Programming (abstract necessary attributes and their types). Write a complete code in C++ programming language.

i. Derive a Computer Science and Mathematics class from student class adding necessary attributes (at least three subjects).

ii. Use these classes in a main function and display the average marks of computer science and mathematics students.

```
#include<iostream>
using namespace std;
class student
{
protected:
   float english, sum, avg;
```

```
public:
   void getstudentdata()
   {
   cout<<"Enter english marks:";
   cin>>english;
   }
};
class computer : public student
{
  float IT, cprog, networks;
public:
   void getcomputerdata()
   {
   cout<<"Enter marks in IT:";
   cin>>IT;
   cout<<"Enter marks in cprog:";
   cin>>cprog;
   cout<<"Enter marks in networks:";
   cin>>networks;
}
   void average()
   {    sum=english+IT+cprog+networks;
   avg=sum/4;
   cout<<"Average marks is"<<avg;
   }
};
class mathematics :public student
```

```
{
   float calculus, stat, algebra;
public:
   void getmathdata()
   {
   cout<<"Enter marks in calculus:";
   cin>>calculus;
   cout<<"Enter marks in statistics:";
   cin>>stat;
   cout<<"Enter marks in Linear Algebra:";
   cin>>algebra;
   }
   void average()
   {    sum=english+calculus+stat+algebra;
   avg=sum/4;
   cout<<"Average marks is"<<avg;
   }
};

int main()
{
computer C;
mathematics M;
cout<<"Enter marks of computer students:";
C.getstudentdata();
C.getcomputerdata();
cout<<"Enter marks of mathematics student:";
```

```
M.getstudentdata();
M.getmathdata();
C.average();
M.average();
}
```

Define a Clock class (with necessary constructor and member functions) in OOP (abstract necessary attributes and their types). Write a complete code in C++ programming language.

i. Derive Wall_Clock class from Clock class adding necessary attributes.

ii. Create two objects of Wall_Clock class with all initial state to empty.

```
#include<iostream>
#include<string.h>
using namespace std;
class clock
{
protected:
char model_no[10];
float price;
char manufacturer[50];
public:
void getclockdata()
{
cout<<"Enter clock manufacturer:"<<endl;
cin>>manufacturer;
cout<<"Enter model number:"<<endl;
cin>>model_no;
cout<<"Enter price:"<<endl;
```

```
cin>>price;
}
void clockdisplay()
{
cout<<"Model number="<<model_no<<endl;
cout<<"Manufacturer="<<manufacturer<<endl;
cout<<"Price="<<price<<endl;
}
};
class wall_clock: public clock
{
int hr, min, sec;
public:
wall_clock()
{
strcpy(model_no,"");
strcpy(manufacturer,"");
price=0.0;
hr=0;
min=0;
sec=0;
}
void getwallclockdata()
{
cout<<"Enter hour, minute and seconds:"<<endl;
cin>>hr>>min>>sec;
}
```

```
void wallclockdisplay()
{
cout<<"Time="<<hr<<":"<<min<<":"<<sec<<endl;
}
};
int main()
{
wall_clock W1, W2;
cout<<"Enter data for W1:"<<endl;
W1.getclockdata();
W1.getwallclockdata();
cout<<"Value of W1:"<<endl;
W1.clockdisplay();
W1.wallclockdisplay();
cout<<"Enter data for W2:"<<endl;
W2.getclockdata();
W2.getwallclockdata();
cout<<"Value of W2:"<<endl;
W2.clockdisplay();
W2.wallclockdisplay();
}
```
