(PRACTICE QUESTIONS)

(MADE BY ALI AKBER)

(BSCS 2ND SS1)

(MULTIPLE INHERITANCE)

We want to store the information of different vehicles. Create a class named Vehicle with two data member named mileage and price. Create its two subclasses

***Car with data members to store ownership cost, warranty (by years), seating capacity and fuel type (diesel or petrol).**

***Bike with data members to store the number of cylinders, number of gears, cooling type(air, liquid or oil), wheel type(alloys or spokes) and fuel tank size(in inches).**

Override them too .

Make another two subclasses Audi and Ford of Car, each having a data member to store the model type. Next, make two subclasses Honda And Yahma, each having a data member to store the make-type.

Now, store and print the information of an Audi and a Ford car (i.e. model type, ownership cost, warranty, seating capacity, fuel type, mileage and price.) Do the same for a Honda and Yahma bike.

```
/// VEHICLE CLASS
/// MULTIPLE INHERITANCE.
/// ALI AKBER R036
#include<iostream>
#include<string.h>
using namespace std;
class Vehicle{
    private:
    double mileage;
    float price;
```

```
    public:
    Vehicle():mileage(0),price(0){}    //no arg constructor.
    Vehicle(double mile,float p):mileage(mile),price(p){}    //two arg constructor.
    void getdata(){
        cout<<"Enter the distance covered by the vehicle:"<<endl;
        cin>>mileage;
        cout<<"Enter the price of the vehicle:"<<endl;
        cin>>price;
    }
    void showdata(){
        cout<<"Distance covered by the vehicle:"<<mileage<<endl;
        cout<<"Price of the vehicle:"<<price<<endl;
    }
};
class Car: private Vehicle{
    private:
    float ownership_cost;
    int warrenty;
    int capacity;
    string fuel;
    public:
    Car():Vehicle(),ownership_cost(0),warrenty(0),capacity(0),fuel(""){} //no arg constructor.
    Car(double mile,float p,float own,int war,int cap,string
f):Vehicle(mile,p),ownership_cost(own),warrenty(war),capacity(cap),fuel(f){}
    //multiple arg constructor.
    void getdata(){
```

```
            Vehicle::getdata();
        cout<<"Enter The price to purchase the car:"<<endl;
    cin>>ownership_cost;
    cout<<"Enter The warrenty of the car(years):"<<endl;
    cin>>warrenty;
    cout<<"Enter The seating capacity of the car:"<<endl;
    cin>>capacity;
    cout<<"Enter The fuel type of the car:"<<endl;
  cin.ignore();
    getline(cin,fuel);
    }
    void showdata(){
            Vehicle::showdata();
        cout<<"Price of the car is:"<<ownership_cost<<endl;
        cout<<"Warrenty of the car is:"<<warrenty<<endl;
        cout<<"Seating Capacity of the car is:"<<capacity<<endl;
        cout<<"Fuel type of the car is:"<<fuel<<endl;
    }
};
class Bike: private Vehicle{
    private:
    int cylinders;
    int gears;
    string cooling;
    string wheel;
    float fuel_tank;
```

```
    public:
    Bike():Vehicle(),cylinders(0),gears(0),cooling(""),wheel(""),fuel_tank(0){} //no arg constructor.
   Bike(double mile,float p,int c,int g,string cool,string w,float
ft):Vehicle(mile,p),cylinders(c),gears(g),cooling(cool),wheel(w),fuel_tank(ft){} //no arg constructor.
    //multiple arg constructor.
    void getdata(){
        Vehicle::getdata();
     cout<<"Enter The cylinders the bike:"<<endl;
    cin>>cylinders;
    cout<<"Enter The gears of the bike:"<<endl;
    cin>>gears;
    cin.ignore();
    cout<<"Enter The cooling type of the bike:"<<endl;
    cin>>cooling;
    cin.ignore();
    cout<<"Enter The wheel type of the bike:"<<endl;
    cin>>wheel;
    cout<<"Enter The fuel tank size of the bike:"<<endl;
    cin>>fuel_tank;
    }
    void showdata(){
          Vehicle::showdata();
        cout<<"Cylinders of the bike are:"<<cylinders<<endl;
        cout<<"Gears of the bike are:"<<gears<<endl;
        cout<<"Cooling type of the bike is:"<<cooling<<endl;
        cout<<"Wheel type of the bike is:"<<wheel<<endl;
```

```
        cout<<"Fuel tank size     of the bike is:"<<fuel_tank<<endl;
    }
};
class Audi: private Car{
    private:
    string model;
    public:
    Audi():Car(),model(""){}    //no arg constructor.
    Audi(double mile,float p,float own, int war, int cap, string f, string mo) : Car(mile,p,own, war, cap, f), model(mo) {} //multiple arg constructor.
    void getdata(){
        Car::getdata();
        cout<<"Enter the model of audi:"<<endl;
        cin>>model;
    }
    void showdata(){
        Car::showdata();
        cout<<"Model of audi is"<<model<<endl;
    }
};
class Ford: private Car{
    private:
    string model;
    public:
    Ford():Car(),model(""){}    //no arg constructor.
    Ford(double mile,float p,float own,int war,int cap,string f,string mo):Car(mile,p,own,war,cap,f),model(mo){}    //multiple arg constructor.
```

```
    void getdata(){
        Car::getdata();
        cout<<"Enter the model of Ford:"<<endl;
        cin>>model;
    }
    void showdata(){
        Car::showdata();
        cout<<"Model of Ford is"<<model<<endl;
    }
};
class Honda: private Bike{
   private:
   string make_type;
   public:
   Honda():Bike(),make_type(""){}    //no arg constructor.
  Honda(double mile,float p,int c,int g,string cool,string w,float ft,string
mk):Bike(mile,p,c,g,w,cool,ft),make_type(mk){}    //multiple arg constructor.
  void getdata(){
        Bike::getdata();
        cout<<"Enter the make type of:"<<endl;
        cin>>make_type;
    }
    void showdata(){
        Bike::showdata();
        cout<<"Makeup type of Honda is"<<make_type<<endl;
    }
```

```
};
class Yahma: private Bike{
   private:
   string make_type;
   public:
   Yahma():Bike(),make_type(""){}    //no arg constructor.
  Yahma(double mile,float p,int c,int g,string cool,string w,float ft,string
mk):Bike(mile,p,c,g,w,cool,ft),make_type(mk){}    //multiple arg constructor.
  void getdata(){
          Bike::getdata();
          cout<<"Enter the make type of:"<<endl;
          cin>>make_type;
     }
     void showdata(){
          Bike::showdata();
          cout<<"Makeup type of Yahma is"<<make_type<<endl;
     }
};
int main(){
Vehicle v;
cout<<"Enter the details of the vehicle:"<<endl;
v.getdata();
cout<<"Details of the vehicle is:"<<endl;
v.showdata();
cout<<endl;
```

```
Car c;
cout<<"Enter the details of the car:"<<endl;
c.getdata();
cout<<"Details of the car is:"<<endl;
c.showdata();
cout<<endl;

Bike b;
cout<<"Enter the details of the bike:"<<endl;
b.getdata();
cout<<"Details of the bike is:"<<endl;
b.showdata();
cout<<endl;

Audi a;
cout<<"Enter the details of the audi:"<<endl;
a.getdata();
cout<<"Details of the audi is:"<<endl;
a.showdata();
cout<<endl;

Ford f;
cout<<"Enter the details of the Ford:"<<endl;
f.getdata();
cout<<"Details of the ford is:"<<endl;
f.showdata();
```

```
cout<<endl;


Honda h;
cout<<"Enter the details of the Honda:"<<endl;
h.getdata();
cout<<"Details of the Honda is:"<<endl;
h.showdata();
cout<<endl;


Yahma y;
cout<<"Enter the details of the Yahma:"<<endl;
y.getdata();
cout<<"Details of the Yahma is:"<<endl;
y.showdata();
cout<<endl;
    return 0;
}
```

(AGGREGATION)

```
/// VEHICLE CLASS
/// WITH AGGREGATION.
/// ALI AKBER R036
#include<iostream>
#include<string.h>
using namespace std;
class Vehicle{
    private:
```

```
    double mileage;
    float price;
    public:
    Vehicle():mileage(0),price(0){}    //no arg constructor.
    Vehicle(double mile,float p):mileage(mile),price(p){}    //two arg constructor.
    void getdata(){
        cout<<"Enter the distance covered by the vehicle:"<<endl;
        cin>>mileage;
        cout<<"Enter the price of the vehicle:"<<endl;
        cin>>price;
    }
    void showdata(){
        cout<<"Distance covered by the vehicle:"<<mileage<<endl;
        cout<<"Price of the vehicle:"<<price<<endl;
    }
};
class Car{
    private:
    float ownership_cost;
    int warrenty;
    int capacity;
    string fuel;
    Vehicle ve;
    public:
    void getdata(){
          ve.getdata();
```

```
        cout<<"Enter The price to purchase the car:"<<endl;
    cin>>ownership_cost;
    cout<<"Enter The warrenty of the car(years):"<<endl;
    cin>>warrenty;
    cout<<"Enter The seating capacity of the car:"<<endl;
    cin>>capacity;
    cout<<"Enter The fuel type of the car:"<<endl;
  cin.ignore();
    getline(cin,fuel);
    }
    void showdata(){
          ve.showdata();
        cout<<"Price of the car is:"<<ownership_cost<<endl;
        cout<<"Warrenty of the car is:"<<warrenty<<endl;
        cout<<"Seating Capacity of the car is:"<<capacity<<endl;
        cout<<"Fuel type of the car is:"<<fuel<<endl;
    }
};
class Bike{
    private:
    int cylinders;
    int gears;
    string cooling;
    string wheel;
    float fuel_tank;
    Vehicle ve;
```

```
    public:
    void getdata(){
        ve.getdata();
      cout<<"Enter The cylinders the bike:"<<endl;
    cin>>cylinders;
    cout<<"Enter The gears of the bike:"<<endl;
    cin>>gears;
    cin.ignore();
    cout<<"Enter The cooling type of the bike:"<<endl;
    cin>>cooling;
    cin.ignore();
    cout<<"Enter The wheel type of the bike:"<<endl;
    cin>>wheel;
    cout<<"Enter The fuel tank size of the bike:"<<endl;
    cin>>fuel_tank;
    }
    void showdata(){
          ve.showdata();
        cout<<"Cylinders of the bike are:"<<cylinders<<endl;
        cout<<"Gears of the bike are:"<<gears<<endl;
        cout<<"Cooling type of the bike is:"<<cooling<<endl;
        cout<<"Wheel type of the bike is:"<<wheel<<endl;
        cout<<"Fuel tank size    of the bike is:"<<fuel_tank<<endl;
    }
};
class Audi{
```

```
    private:
    string model;
    Car cr;
    public:
    void getdata(){
        cr.getdata();
        cout<<"Enter the model of audi:"<<endl;
        cin>>model;
    }
    void showdata(){
        cr.showdata();
        cout<<"Model of audi is"<<model<<endl;
    }
};
class Ford{
    private:
    string model;
Car cr;
    public:
    void getdata(){
        cr.getdata();
        cout<<"Enter the model of Ford:"<<endl;
        cin>>model;
    }
    void showdata(){
        cr.showdata();
```

```
        cout<<"Model of Ford is"<<model<<endl;
    }
};
class Honda{
   private:
   string make_type;
Bike bk;
   public:
   void getdata(){
        bk.getdata();
        cout<<"Enter the make type of:"<<endl;
        cin>>make_type;
    }
     void showdata(){
        bk.showdata();
        cout<<"Makeup type of Honda is"<<make_type<<endl;
    }
};
class Yahma: private Bike{
   private:
   string make_type;
Bike bk;
   public:
  void getdata(){
        bk.getdata();
        cout<<"Enter the make type of:"<<endl;
```

```
        cin>>make_type;
    }
    void showdata(){
        bk.showdata();
        cout<<"Makeup type of Yahma is"<<make_type<<endl;
    }
};
int main(){
Vehicle v;
cout<<"Enter the details of the vehicle:"<<endl;
v.getdata();
cout<<"Details of the vehicle is:"<<endl;
v.showdata();
cout<<endl;

Car c;
cout<<"Enter the details of the car:"<<endl;
c.getdata();
cout<<"Details of the car is:"<<endl;
c.showdata();
cout<<endl;

Bike b;
cout<<"Enter the details of the bike:"<<endl;
b.getdata();
cout<<"Details of the bike is:"<<endl;
```

```
b.showdata();
cout<<endl;

Audi a;
cout<<"Enter the details of the audi:"<<endl;
a.getdata();
cout<<"Details of the audi is:"<<endl;
a.showdata();
cout<<endl;

Ford f;
cout<<"Enter the details of the Ford:"<<endl;
f.getdata();
cout<<"Details of the ford is:"<<endl;
f.showdata();
cout<<endl;

Honda h;
cout<<"Enter the details of the Honda:"<<endl;
h.getdata();
cout<<"Details of the Honda is:"<<endl;
h.showdata();
cout<<endl;

Yahma y;
cout<<"Enter the details of the Yahma:"<<endl;
```

```
y.getdata();
cout<<"Details of the Yahma is:"<<endl;
y.showdata();
cout<<endl;
      return 0;
}
```

(PURE VIRTUAL FUNCTION)

```
/// FRUIT CLASS
/// MULTIPLE INHERITANCE.
/// PURE VIRTUAL FUNCTION.
/// VIRTUAL DESTRUCTOR.
/// USING AN ARRAY OF POINTER.
/// ALI AKBER R036
#include<iostream>
#include<string.h>
using namespace std;
class Fruit{
public:
virtual void show()=0;    ///pure virtual function
virtual ~Fruit(){
      cout<<"I am a Fruit:"<<endl;
}      ///virtual destructor.
};
class Apple: public Fruit{
  void show(){
      cout<<"I am an Apple:"<<endl;
```

```
  }
virtual ~Apple(){
      cout<<"I was an Apple:"<<endl;
}      ///virtual destructor.
};
class Banana: public Fruit{
  void show(){
      cout<<"I am a Banana:"<<endl;
  }
virtual ~Banana(){
      cout<<"I was a Banana:"<<endl;
}      ///virtual destructor.
};
class Orange: public Fruit{
  void show(){
      cout<<"I am a Orange:"<<endl;
  }
virtual ~Orange(){
      cout<<"I was an Orange:"<<endl;
}      ///virtual destructor.
};
class Mango: public Fruit{
  void show(){
      cout<<"I am a Mango:"<<endl;
  }
virtual ~Mango(){
```

```
      cout<<"I was a Mango:"<<endl;
}      ///virtual destructor.
};
class Grapes: public Fruit{
  void show(){
      cout<<"I am the Grapes:"<<endl;
  }
virtual ~Grapes(){
      cout<<"I was the Grapes:"<<endl;
}      ///virtual destructor.
};
class Guava: public Fruit{
  void show(){
      cout<<"I am the gauva:"<<endl;
  }
virtual ~Guava(){
      cout<<"I was the Gauva:"<<endl;
}      ///virtual destructor.
};
class Melon: public Fruit{
  void show(){
      cout<<"I am the Melon:"<<endl;
  }
virtual ~Melon(){
      cout<<"I was the Melon:"<<endl;
}      ///virtual destructor.
```

```
};
class WaterMelon: public Fruit{
  void show(){
      cout<<"I am the WaterMelon:"<<endl;
  }
virtual ~WaterMelon(){
      cout<<"I was the WaterMelon:"<<endl;
}      ///virtual destructor.
};
class Apricot: public Fruit{
  void show(){
      cout<<"I am the Apricot:"<<endl;
  }
virtual ~Apricot(){
      cout<<"I was the Apricot:"<<endl;
}      ///virtual destructor.
};
class Pineapple: public Fruit{
  void show(){
      cout<<"I am the Pineapple:"<<endl;
  }
virtual ~Pineapple(){
      cout<<"I was the Pineapple:"<<endl;
}      ///virtual destructor.
};
int main(){
```

```
Fruit* ptr[100];
int n=0; int choice;
char opt='y';
do{
  cout<<"1-APPLE"<<endl;
  cout<<"2-BANANA"<<endl;
  cout<<"3-ORANGE"<<endl;
  cout<<"4-MANGO"<<endl;
  cout<<"5-GRAPES"<<endl;
  cout<<"6-GUAVA"<<endl;
  cout<<"7-MELON"<<endl;
  cout<<"8-WATERMELON"<<endl;
  cout<<"9-APRICOT"<<endl;
  cout<<"10-PINEAPPLE"<<endl;

  cout<<"Enter option:"<<endl;
  cin>>choice;
  switch(choice){
    case 1:
    ptr[n++]=new Apple();
    break;
  case 2:
    ptr[n++]=new Banana();
    break;
    case 3:
    ptr[n++]=new Orange();
```

```
    break;
    case 4:
    ptr[n++]=new Mango();
    break;
    case 5:
    ptr[n++]=new Grapes();
    break;
    case 6:
    ptr[n++]=new Guava();
    break;
    case 7:
    ptr[n++]=new Melon();
    break;
    case 8:
    ptr[n++]=new WaterMelon();
    break;
    case 9:
    ptr[n++]=new Apricot();
    break;
    case 10:
    ptr[n++]=new Pineapple();
    break;
    default:
    cout<<"Invalid choice:"<<endl;
}
cout<<"Do you want to continue(y/n)"<<endl;
```

```
  cin>>opt;
  }
  while(opt=='Y' || opt=='y');
cout<<endl;
for (int i=0;i<n;i++){
      ptr[i]->show();
      //delete ptr[i];
}
      return 0;
}
```

(VIRTUAL AND PURE VIRTUAL FUNCTIONS)

```
///PAST PAPER QUESTION.
///VIRTUAL AND PURE VIRTUAL FUNCTION.
/// ALI AKBER R036
#include<iostream>
using namespace std;
class Shape{
      protected:
      double length;
      double width;
      public:
      Shape():length(0),width(0){}     //no arg constructor.
      Shape(double l,double w):length(l),width(w){}     //two arg constructor.
      virtual void getdata() {
                  cout << "Enter the length of the shape:" << endl;
                  cin >> length;
```

```
            cout << "Enter the width of the shape:" << endl;
            cin >> width;
        }
    virtual double showdata()=0;      //pure virtual function.
};
class Rectangle: public Shape{
    public:
    Rectangle():Shape(){}    //no arg constructor.
    Rectangle(double l,double w):Shape(l,w){}    //multiple arg constructor.
      double showdata(){
        return length*width;
      }
};
class Triangle: public Shape{
    public:
    Triangle():Shape(){}    //no arg constructor.
    Triangle(double l,double w):Shape(l,w){}    //multiple arg constructor.
      double showdata(){
        return 0.5*length*width;
      }
};
int main(){
    Shape* ptr[5];
    int n=0;
    char choice;
    char opt='y';
```

```
    do{
        cout<<"1-RECTANGLE"<<endl;
        cout<<"2-TRIANGLE"<<endl;

        cout<<"Enter Your choice"<<endl;
        cin>>choice;
        if (choice=='R' || choice=='r'){
            ptr[n]= new Rectangle();
            ptr[n++]->getdata();
        }
            else if (choice=='T' || choice=='t'){
            ptr[n]= new Triangle();
            ptr[n++]->getdata();
            }
            else {
cout<<"INVLALID OPTION!!!"<<endl;
            }
            cout<<"Do you want to continue(y/n)"<<endl;
            cin>>opt;
        }
        while(opt=='Y' || opt=='y');
        cout<<endl;
        for (int j=0; j<n; j++){
            double area = ptr[j]->showdata();
      cout << "Area of Shape " << j + 1 << ": " << area << endl;
        }
```

```
        for (int j = 0; j < n; j++) {
            delete ptr[j]; // Deallocate memory for each dynamically created object.
       }
    return 0;
}
```

(FRIEND FUNCTIONS AND THIS POINTER)

```
///FRIEND FUNCTION
/// INSERTION AND EXTRACTION OPERATOR OVERLOADING.
/// STUDENT CLASS
/// THIS POINTER
/// ALI AKBER R036
#include<iostream>
#include<string.h>
using namespace std;
class Student{
    protected:
    int rollno;
    string name;
    public:
    Student():rollno(0),name(""){
        cout<<"I am no argument constructor of student class:"<<this<<endl;    //no arg constructor.
    }
    Student(int rn,string na){
        this->rollno=rn;
        this->name=na;
        cout<<"I am two argument constructor of student class:"<<this<<endl;
```

```
    }
    friend ostream& operator<<(ostream&,Student&);
    friend istream& operator>>(istream&,Student&);
};
 istream& operator>>(istream& in,Student& s){
 cout<<"Enter the student roll no:"<<endl;
 in>>s.rollno;
 cout<<"Enter the student name:"<<endl;
 in.ignore();
 getline(in,s.name);
 return in;
}
ostream& operator<<(ostream& out,Student& s){
 out<<"Student roll no is:"<<s.rollno<<endl;;
 out<<"Student name is:"<<s.name<<endl;
 return out;
}
class MajorSubject: public Student{
private:
int oop;
int dld;
public:
MajorSubject():Student(),oop(0),dld(0){
    cout<<"I am no argument constructor of MajorSubject class"<<this<<endl;
};    //no arg constructor.
MajorSubject(int o,int d,int rn,string na):Student(rn,na),oop(o),dld(d){
```

```
    cout<<"I am three argument constructor of student class"<<this<<endl;
};     //three arg constructor.
    friend ostream& operator<<(ostream&,MajorSubject&);
    friend istream& operator>>(istream&,MajorSubject&);
    int MajorSubjects_marks(){
        return oop+dld;
    }
};
istream& operator>>(istream& in,MajorSubject& maj){
  cout<<"Enter the student marks in oop:"<<endl;
  in>>maj.oop;
  cout<<"Enter the student marks in dld:"<<endl;
  in>>maj.dld;
  return in;
}
ostream& operator<<(ostream& out,MajorSubject& maj){
  out<<"Student marks in oop are:"<<maj.oop<<endl;
  out<<"Student marks in dld are:"<<maj.dld<<endl;
  return out;
}

class MinorSubject: public Student{
private:
int communication_skills;
int stats;
public:
```

```
MinorSubject():Student(),communication_skills(0),stats(0){
    cout<<"I am no argument constructor of MinorSubject class:"<<this<<endl;
};    //no arg constructor.
MinorSubject(int cs,int s,int rn,string na):Student(rn,na),communication_skills(cs),stats(s){
    cout<<"I am three argument constructor of student class:"<<this<<endl;
};    //three arg constructor.
    friend ostream& operator<<(ostream&,MinorSubject&);
    friend istream& operator>>(istream&,MinorSubject&);
     int MinorSubjects_marks(){
         return communication_skills+stats;
    }
};
istream& operator>>(istream& in,MinorSubject& min){
  cout<<"Enter the student marks in Communication Skills:"<<endl;
  in>>min.communication_skills;
  cout<<"Enter the student marks in Stats:"<<endl;
  in>>min.stats;
  return in;
}
ostream& operator<<(ostream& out,MinorSubject& min){
  out<<"Student marks in Communication Skills are:"<<min.communication_skills<<endl;
  out<<"Student marks in Stats are:"<<min.stats<<endl;
  return out;
}
int main(){
    Student s;
```

```
cout<<"Enter Student details:"<<endl;
cin>>s;
cout<<"Student Details is as follows:"<<endl;
cout<<s;
cout<<endl;

  MajorSubject mj;
cout<<"Enter Student marks in Major Subjects details:"<<endl;
cin>>mj;
  cout<<"Student Details in Major Subjects is as follows:"<<endl;
cout<<mj;
cout<<endl;
int od=0;;
od=mj.MajorSubjects_marks();
cout<<"Major Subjects marks are:"<<od<<endl;

  MinorSubject mi;
cout<<"Enter Student marks in Minor Subjects details:"<<endl;
cin>>mi;
  cout<<"Student Details in Minor Subjects is as follows:"<<endl;
cout<<mi;
cout<<endl;
int cs=0;;
cs=mi.MinorSubjects_marks();
cout<<"Minor Subject Marks are:"<<cs<<endl;
```

```
    float average=0.0f;
    average=((od+cs)*100.0f)/400;
    cout<<"Average total marks are:"<<average<<endl;
    return 0;
}
```

(BUILT IN EXCEPTION HANDLING)

```
/// VEHICLE CLASS
/// MULTIPLE INHERITANCE.
/// FRIEND FUNCTION.
/// INSERTION AND EXTRACTION OPERATOR OVERLOADING.
/// BUILT IN EXCEPTION HANDLING.
/// ALI AKBER R036
#include<iostream>
#include<stdexcept>
#include<string.h>
using namespace std;
class Vehicle{
    private:
    double mileage;
    float price;
    public:
    Vehicle():mileage(0),price(0){}    //no arg constructor.
    Vehicle(double mile,float p):mileage(mile),price(p){}    //two arg constructor.
    friend istream& operator>>(istream& ,Vehicle&);
    friend ostream& operator<<(ostream& ,Vehicle&);
};
```

```
istream& operator>>(istream& in,Vehicle& v){
        cout<<"Enter the distance covered by the vehicle:"<<endl;
        in>>v.mileage;
          if(cin.fail()){
        throw runtime_error("INPUT MUST BE INTEGER..");
    }
        cout<<"Enter the price of the vehicle:"<<endl;
        in>>v.price;
        return in;
}
ostream& operator<<(ostream& out,Vehicle& v){
          out<<"Distance covered by the vehicle:"<<v.mileage<<endl;
          out<<"Price of the vehicle:"<<v.price<<endl;
          return out;
}
class Car: private Vehicle{
    private:
    float ownership_cost;
    int warrenty;
    int capacity;
    string fuel;
    public:
    Car():Vehicle(),ownership_cost(0),warrenty(0),capacity(0),fuel(""){} //no arg constructor.
    Car(double mile,float p,float own,int war,int cap,string
f):Vehicle(mile,p),ownership_cost(own),warrenty(war),capacity(cap),fuel(f){}
    //multiple arg constructor.
```

```
        friend istream& operator>>(istream&,Car&);
      friend ostream& operator<<(ostream&,Car&);


};
    istream& operator>>(istream& in,Car& c){
    cout<<"Enter The price to purchase the car:"<<endl;
      in>>c.ownership_cost;
        if(cin.fail()){
            throw runtime_error("INPUT MUST BE INTEGER..");
      }
      cout<<"Enter The warrenty of the car(years):"<<endl;
      in>>c.warrenty;
      cout<<"Enter The seating capacity of the car:"<<endl;
      in>>c.capacity;
      cout<<"Enter The fuel type of the car:"<<endl;
  in.ignore();
      getline(in,c.fuel);
      return in;
  }
  ostream& operator<<(ostream& out,Car& c){
            out<<"Price of the car is:"<<c.ownership_cost<<endl;
            out<<"Warrenty of the car is:"<<c.warrenty<<endl;
            out<<"Seating Capacity of the car is:"<<c.capacity<<endl;
            out<<"Fuel type of the car is:"<<c.fuel<<endl;
            return out;
  }
```

```
class Bike: private Vehicle{
      private:
      int cylinders;
      int gears;
      string cooling;
      string wheel;
      float fuel_tank;
      public:
      Bike():Vehicle(),cylinders(0),gears(0),cooling(""),wheel(""),fuel_tank(0){} //no arg constructor.
    Bike(double mile,float p,int c,int g,string cool,string w,float
ft):Vehicle(mile,p),cylinders(c),gears(g),cooling(cool),wheel(w),fuel_tank(ft){} //no arg constructor.
      //multiple arg constructor.
        friend istream& operator>>(istream&,Bike&);
      friend ostream& operator<<(ostream&,Bike&);
};
istream& operator>>(istream& in,Bike& b){
    cout<<"Enter The cylinders the bike:"<<endl;
      in>>b.cylinders;
        if(cin.fail()){
            throw runtime_error("INPUT MUST BE INTEGER..");
      }
      cout<<"Enter The gears of the bike:"<<endl;
      in>>b.gears;
      in.ignore();
      cout<<"Enter The cooling type of the bike:"<<endl;
      in>>b.cooling;
```

```
    in.ignore();
    cout<<"Enter The wheel type of the bike:"<<endl;
    in>>b.wheel;
    cout<<"Enter The fuel tank size of the bike:"<<endl;
    in>>b.fuel_tank;
    return in;
}
ostream& operator<<(ostream& out,Bike& b){
        out<<"Cylinders of the bike are:"<<b.cylinders<<endl;
        out<<"Gears of the bike are:"<<b.gears<<endl;
        out<<"Cooling type of the bike is:"<<b.cooling<<endl;
        out<<"Wheel type of the bike is:"<<b.wheel<<endl;
        out<<"Fuel tank size    of the bike is:"<<b.fuel_tank<<endl;
        return out;
}
class Audi: private Car{
    private:
    string model;
    public:
    Audi():Car(),model(""){}    //no arg constructor.
    Audi(double mile,float p,float own, int war, int cap, string f, string mo) : Car(mile,p,own, war, cap, f),
model(mo) {} //multiple arg constructor.
    friend istream& operator>>(istream&,Audi&);
    friend ostream& operator<<(ostream&,Audi&);
};
istream& operator>>(istream& in,Audi& a){
```

```
        cout<<"Enter the model of audi:"<<endl;
        in>>a.model;
          if(cin.fail()){
        throw runtime_error("INPUT MUST BE INTEGER..");
    }
        return in;
}
ostream& operator<<(ostream& out,Audi& a){
out<<"Model of audi is"<<a.model<<endl;
return out;
}
class Ford: private Car{
    private:
    string model;
    public:
    Ford():Car(),model(""){}    //no arg constructor.
    Ford(double mile,float p,float own,int war,int cap,string f,string
mo):Car(mile,p,own,war,cap,f),model(mo){}    //multiple arg constructor.
    friend istream& operator>>(istream&,Ford&);
    friend ostream& operator<<(ostream&,Ford&);
};
istream& operator>>(istream& in,Ford& f){
      cout<<"Enter the model of Ford:"<<endl;
    in>>f.model;
      if(cin.fail()){
        throw runtime_error("INPUT MUST BE INTEGER..");
```

```
    }
    return in;
}
ostream& operator<<(ostream& out,Ford& f){
  out<<"Model of Ford is"<<f.model<<endl;
  return out;
}
class Honda: private Bike{
   private:
   string make_type;
   public:
   Honda():Bike(),make_type(""){}    //no arg constructor.
  Honda(double mile,float p,int c,int g,string cool,string w,float ft,string
mk):Bike(mile,p,c,g,w,cool,ft),make_type(mk){}    //multiple arg constructor.
  friend istream& operator>>(istream&,Honda&);
friend ostream& operator<<(ostream&,Honda&);
};
istream& operator>>(istream& in,Honda& h){
  cout<<"Enter the make type of:"<<endl;
          in>>h.make_type;
            if(cin.fail()){
          throw runtime_error("INPUT MUST BE INTEGER..");
    }
          return in;
}
ostream& operator<<(ostream& out,Honda& h){
```

```
  out<<"Makeup type of Honda is"<<h.make_type<<endl;
  return out;
}
class Yahma: private Bike{
    private:
    string make_type;
    public:
    Yahma():Bike(),make_type(""){}    //no arg constructor.
  Yahma(double mile,float p,int c,int g,string cool,string w,float ft,string
mk):Bike(mile,p,c,g,w,cool,ft),make_type(mk){}    //multiple arg constructor.
        friend istream& operator>>(istream&,Yahma&);
friend ostream& operator<<(ostream&,Yahma&);
};
istream& operator>>(istream& in,Yahma& y){
            cout<<"Enter the make type of:"<<endl;
            in>>y.make_type;
              if(cin.fail()){
            throw runtime_error("INPUT MUST BE INTEGER..");
      }
            return in;
}
ostream& operator<<(ostream& out,Yahma& y){
out<<"Makeup type of Yahma is"<<y.make_type<<endl;
return out;
}
int main(){
```

```
    try{
Vehicle v;
cout<<"Enter the details of the vehicle:"<<endl;
cin>>v;
cout<<"Details of the vehicle is:"<<endl;
cout<<v;
cout<<endl;
    }
catch(runtime_error& e){
    cout<<"ERROR....."<<e.what()<<endl;
}
try{
Car c;
cout<<"Enter the details of the car:"<<endl;
cin>>c;
cout<<"Details of the car is:"<<endl;
cout<<c;
cout<<endl;
}
catch(runtime_error& e){
    cout<<"ERROR....."<<e.what()<<endl;
}
try{
Bike b;
cout<<"Enter the details of the bike:"<<endl;
cin>>b;
```

```
cout<<"Details of the bike is:"<<endl;
cout<<b;
cout<<endl;
}
catch(runtime_error& e){
    cout<<"ERROR....."<<e.what()<<endl;
}
try{
Audi a;
cout<<"Enter the details of the audi:"<<endl;
cin>>a;
cout<<"Details of the audi is:"<<endl;
cout<<a;
cout<<endl;
}
catch(runtime_error& e){
    cout<<"ERROR....."<<e.what()<<endl;
}
try{
Ford f;
cout<<"Enter the details of the Ford:"<<endl;
cin>>f;
cout<<"Details of the ford is:"<<endl;
cout<<f;
cout<<endl;
}
```

```
catch(runtime_error& e){
    cout<<"ERROR....."<<e.what()<<endl;
}
try{
Honda h;
cout<<"Enter the details of the Honda:"<<endl;
cin>>h;
cout<<"Details of the Honda is:"<<endl;
cout<<h;
cout<<endl;
}
catch(runtime_error& e){
    cout<<"ERROR....."<<e.what()<<endl;
}
try{
Yahma y;
cout<<"Enter the details of the Yahma:"<<endl;
cin>>y;
cout<<"Details of the Yahma is:"<<endl;
cout<<y;
cout<<endl;
}
catch(runtime_error& e){
    cout<<"ERROR....."<<e.what()<<endl;
}
    return 0;
```

```
}
```

(USER DEFINED EXCEPTION HANDLING)

```
/// SOCIAL MEDIA CLASS

/// MULTIPLE INHERITANCE.

/// FRIEND FUNCTION.

/// INSERTION AND EXTRACTION OPERATOR OVERLOADING.

/// USER DEFINED EXCEPTION HANDLING.

/// ALI AKBER R036

#include<iostream>

#include<stdexcept>

#include<string.h>

using namespace std;

class SocialMedia_Exception: public runtime_error{

public:

int reminder;

SocialMedia_Exception(string message,int r):reminder(r),runtime_error(message){}

};

class SocialMedia{

private:

int time;

public:

SocialMedia():time(0){} //no arg constructor.

SocialMedia(int t):time(t){} //one arg constructor.

friend istream& operator>>(istream&,SocialMedia&);

friend ostream& operator<<(ostream&,SocialMedia&);

};
```

```
istream& operator>>(istream& in,SocialMedia& sm){
cout<<"Enter the time you spend on social media daily (in hours)"<<endl;
in>>sm.time;
if(sm.time>6){
throw SocialMedia_Exception("YOU SHOULD STOP USING SOCIAL MEDIA NOW!!!",sm.time);
}
return in;
}
ostream& operator<<(ostream& out,SocialMedia& sm){
    out<<"Total time you spend on social media is:"<<sm.time<<endl;
    return out;
}
class Instagram: public SocialMedia{
string scroll;
int hours;
public:
Instagram():SocialMedia(),scroll(""),hours(0){} //no arg constructor.
Instagram(string s,int h,int t):SocialMedia(t),scroll(s),hours(h){} //three arg constructor.
friend istream& operator>>(istream&,Instagram&);
friend ostream& operator<<(ostream&,Instagram&);
};
istream& operator>>(istream& in,Instagram& insta){
cout<<"Which type of content you search on Instagram:"<<endl;
in>>insta.scroll;
cout<<"How many hours you spend on social media:"<<endl;
in>>insta.hours;
```

```
return in;
}
ostream& operator<<(ostream& out,Instagram& insta){
    out<<"The content you searches on Instagram is:"<<insta.scroll<<endl;
    out<<"Time you spend on Instagram is:"<<insta.hours<<endl;
    if(insta.hours>2){
throw SocialMedia_Exception("YOU SHOULD STOP USING INSTAGRAM    NOW!!!",insta.hours);
}
    return out;
}

class Facebook: public SocialMedia{
string watch;
int watch_time;
public:
Facebook():SocialMedia(),watch(""),watch_time(0){} //no arg constructor.
Facebook(string w,int wt,int t):SocialMedia(t),watch(w),watch_time(wt){} //three arg constructor.
friend istream& operator>>(istream&,Facebook&);
friend ostream& operator<<(ostream&,Facebook&);
};
istream& operator>>(istream& in,Facebook& fb){
cout<<"Which type of content you watch on Facebook:"<<endl;
in>>fb.watch;
cout<<"How many hours you spend on Facebook:"<<endl;
in>>fb.watch_time;
return in;
```

```
}
ostream& operator<<(ostream& out,Facebook& fb){
    out<<"The content you watch on Facebook is:"<<fb.watch<<endl;
    out<<"Time you spend on Facebook is:"<<fb.watch_time<<endl;
    if(fb.watch_time>2){
throw SocialMedia_Exception("YOU SHOULD STOP USING FACEBOOK    NOW!!!",fb.watch_time);
}
    return out;
}
int main(){
    try{
        SocialMedia sm;
        cin>>sm;
        cout<<sm;
    }
    catch(SocialMedia_Exception& e){
cout<<"NOTIFICATION....."<<e.what()<<"BECAUSE YOU USED THE SOCIAL MEDIA :"<<e.reminder<<"
HOURS TODAY!!!"<<endl;
    }
     try{
        Instagram insta;
        cin>>insta;
        cout<<insta;
    }
    catch(SocialMedia_Exception& e){
cout<<"NOTIFICATION....."<<e.what()<<"BECAUSE YOU USED THE INSTAGRAM :"<<e.reminder<<"
HOURS TODAY!!!"<<endl;
```

```
    }
     try{
        Facebook fb;
        cin>>fb;
        cout<<fb;
    }
    catch(SocialMedia_Exception& e){
cout<<"NOTIFICATION....."<<e.what()<<"BECAUSE YOU USED THE FACEBOOK :"<<e.reminder<<" HOURS TODAY!!!"<<endl;
    }
    cout<<"MOBILE BATTERY DEAD...."<<endl;
    return 0;
}
```

(PAST PAPER QUESTION)

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

class STRING
{
private:
      char str[50];

public:
      // Default constructor
```

```
    STRING()
    {
        strcpy(str, " ");
    }

    // Constructor with parameter
    STRING(const char s[])
    {
        strcpy(str, s);
        if (strlen(s) == 0)
        {
            throw runtime_error("String cannot be null");
        }
    }

    // Overloaded insertion operator (<<)
    friend ostream &operator<<(ostream &out, const STRING &s);

    // Overloaded extraction operator (>>)
    friend istream &operator>>(istream &in, STRING &s);

    // Overloaded + operator for string concatenation
    friend STRING operator+(const STRING &s1, const STRING &s2);
};

ostream &operator<<(ostream &out, const STRING &s)
```

```
{
	out << s.str;
	return out;
}

istream &operator>>(istream &in, STRING &s)
{
	cout << "Enter your String: ";
	in.getline(s.str, 50);
	if (strlen(s.str) == 0)
	{
		throw runtime_error("String cannot be empty");
	}
	return in;
}

STRING operator+(const STRING &s1, const STRING &s2)
{
	char temp[100];
	strcpy(temp, s1.str);
	strcat(temp, s2.str);
	return STRING(temp);
}

int main()
{
```

```
	try
	{
		STRING s1("Asmaad"), s2, s3;
		cout << endl;
		cin >> s2;
		cout << endl;
		cout << s1;
		cout << endl;
		cout << s1 + s2;
	}
	catch (runtime_error &e)
	{
		cout << "Error: " << e.what() << endl;
	}

	return 0;
}
```