Objective:

- To look into the technical and theoretical aspects of the Polymorphism.

Task 1:

What will be displayed on console for the following code segments?

Q. No. 1.1

```
class IX {
public:  virtual void f()=0;
         virtual void g()=0;
};
Class IY {
public:  virtual void h()=0;
         virtual void show()=0;
};
class concrete: public IX, public IY {
public:  void f() {cout<<"f()";}
         void g() {cout<<"g()";}
         void show()
  {cout<<"show()";}
         void h() {cout<<"h()";}
};
void main() {
        IX* ptr_IX=new concrete;
        IY* ptr_IY=(IY*)(ptr_IX);
        Ptr_IY->show();
}
```

Q. No. 1.2

```
class base {
public:
     virtual void fun() {cout<<"base";}
};
class derive: public base {
private:
    void fun() { cout<<"derive"; }
};
void main() {
        base* ptr=new derive;
        ptr->fun();
}
```

Q. No. 1.3

```
class base1 {
public:
     int a,b;
};
class base2 {
public:
     int c,d;
};
class derive:public base1, public base2 {};
void main() {
        derive obj;
        //Assume: starting address of
       //obj is 100
        cout<<sizeof(obj);
        base1 *ptrb1=&obj;
        base2 *ptrb2=&obj;
        cout<<endl<<ptrb1;
        cout<<endl<<ptrb2;
}
```

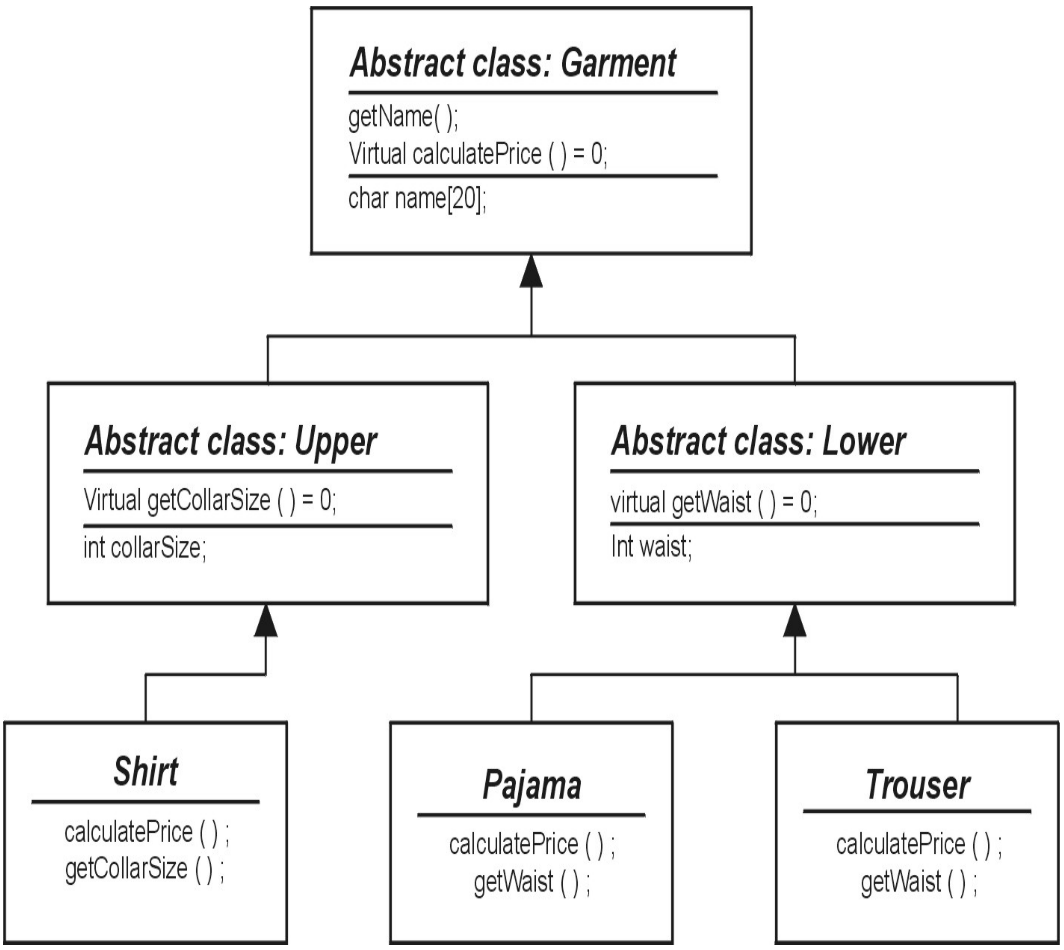
Q. No. 1.4

```
class Pet
{
public:
  void eat()        { cout<<"eat"; }
  void speak()      { cout<<"speak"; }
  void sleep()      { cout<<"sleep"; }
  void sleep(int i) {cout<<"sleep i"; }
};
class Goldfish : Pet {
public:
using Pet::eat;
using Pet::sleep;
};
void main() {
  Goldfish bob;
  bob.eat();
  bob.sleep();
  bob.sleep(1);
  Pet *ptr=(Pet*)&bob;
  Ptr->speak();
}
```

Task-2:

Thanks to: [Aftab Hassan Naqvi]

- Create the classes as shown below, with the indicated functions and attributes for a shopping cart.



Requirements:

- The price is calculated as under,
  - Shirt Price = collar_size*20 + 10;
  - Pajama price = waist*15 + 50;
  - Trouser price = waist*20 + 100;
  - Collar size of shirt = collar_size of upper-1;
  - Waist of pajama = waist – 2;
  - Waist of trouser = waist – 3;

In main(), provide a menu through which user is allowed to add at most 10 items for his\her shopping cart. The user may also remove any item before the final payment is made. (This may be implanted using an array of size 10).

The main menu may have the following options,

- Add an item to shopping cart,
  When this option is selected, the user is shown the list of the available items and is allowed to select one at a time. (at most 10)
- Remove an item from the cart,
  When this option is selected, the user is shown the list of selected items and is allowed to droop on at a time.
- Make the payment,
  When this option is selected, the user is shown total bill and the program exits, “thank you for shopping at our store”.

Task-3:

The goal in this problem is to write classes and a main program to simulate different pricing plans for an internet service provider. This exercise concentrates on using inheritance and polymorphic methods (virtual functions). Suppose you work in the marketing department of the Baltimore On-Line (BOL) service provider. You want to determine which payment plan is better for different customers. The types of plans include the Customer, Hacker, and ChatRoom plans:

Table 1: Different ISP Plans

	Customer	Hacker	ChatRoom
Initial Time (hrs)	4	10	unlimited
Initial Cost (dollars)	10	20	50
Additional Rate (dollars / hr)	4.00	2.50	none
Disk Space Limit (MB)	1	50	unlimited
Charge per connection	none	none	0.10

In the Customer plan, customers get 4 hours of connection time a month for $10.00 and pay a fixed rate of $4.00 an hour for access time over 4 hours. The Hacker plan is designed so that customers who spend more time on-line save money by paying a higher monthly fee of $20.00 for the first ten hours, and $2.50 an hour after that. ChatRoom plan members pay $50.00 for unlimited monthly access and also pay an additional charge of 10 cents for each time they dial up.

Write class definitions for Customer and for the derived sub-classes Hacker, and ChatRoom. Customer should contain data members common to all customers, like the number of hours connected for a month, the customer's name, the amount of disk space the customer requires and so on. Also implement a method for computing a bill, computeBill( ) which uses the appropriate algorithm and data members to compute a monthly charge for each type of customer. In all classes be sure to also include any constructors and other methods that you think are needed. The derived classes should contain any data members and initializations specific to it's payment plan, and a specific implemenation for the computeBill() method which overrides the default method from the Customer class. Demonstrate programmatically which plan is better for the following customers:

- John Dough, a customer spending 6 hours on-line per month who dials-up 35 times and does not host any web pages (uses zero megabytes of disk space).
- Jane Doe, a customer who spends 18 hours on-line per month, who dials-up 75 times and hosts 30 megabytes of web pages and images. (Note: Customer is not a valid option because of the 1 MB Disk space limit).
- Javiar Dinero, a customer who spends 48 hours on-line per month, who dials-up 90 times, and who hosts 10 megabytes of web pages. (Note: Customer is not a valid option because of the 1 MB Disk space limit).