Lab Assignment 2
(Week 1 – Lab C & Lab D)

Q1. You are required to create a class — “Invoice”. This class might be used by a departmental store to represent an invoice for an item sold at the store. An Invoice should include four data members— item number (type string), item description or name (type string), quantity of the item being purchased (type int) and price per item (type int). Your class should have a constructor that initializes the four data members. Provide a set and a get function for each data member. In addition, provide a member function named getInvoiceAmount that calculates the invoice amount (*i.e.*, multiplies the quantity by the price per item), then returns the amount as an int value. The class should be able to give the count of all the objects which are created through copy operation. Write a function that accepts two invoices for different departmental stores and return the maximum quantity out of two invoices. Write a test program that demonstrates class Invoice’s capabilities.

Q2. A grocery shop stores a variety of products which are identified through product_id, prize, name, manufacturing date. A product can be consumable & non consumable. A consumable product in addition is also having the expiry date. A grocery shop is having a no. of customers identified by the name & address. The customers can be members & non-members. The members are having unique membership id while the non-members are having a unique 4 digit mobile no. Grocery shop gives a discount of 20% on the total bill to the members & 2% & 4% on consumable & non consumable goods to the non-members. Use the concept of inheritance & overloading to implement the above scenario. Total discount availed at the end of a day is required to be generated assuming 10 customers shopping for at max 5 products. Display the customer detail that has availed the maximum discount.

Q3. Analyze the sequence of constructor and destructor invocation in following C++ program and obtain the output without running the code

```
class ABC
{
     int x;
     public:
        ABC() { cout<<”1”; }
      ~ABC() { cout<<”2”; }
};
```

```
class KLM
{
     int y;
     ABC O1, *O2;
     public:
        KLM() { cout<<”3”; O2 = new ABC;}
      ~KLM() { cout<<”4”; }
};
```

```
class XYZ
{
     int y; KLM O3, *O4, *O5;
     ABC O6;
     public:
          XYZ() { cout<<”5”; O4 = new KLM;}
        ~XYZ() { cout<<”6”; }
};
```

```
int main() { XYZ *O7; KLM *O8; O7 = new XYZ; O8 = new KLM; delete(O7); return 0; }
```

Q4. Analyze the sequence of constructor invocation in following C++ program and obtain the output without running the code.

```
class A
{
public:
   A(){cout<<"1";}
   A(int x){cout<<"2";}
};
```

```
class B
{
   A O1;
public:
   B(){cout<<"3";}
   B(int x){cout<<"4";}
   B(int x, int y):O1(5) {cout<<"5";}
};
```

```
class C : public B
{
public:
   C(){cout<<"6";}
   C(int x){cout<<"7";}
   C(int x, int y): B(5) {cout<<"8";}
   C(int x, int y, int z): B(5, 6) {cout<<"9";}
};
```

```
int main() { B O2, O3(5), O4(5,6); return 0; }
```

Q5. Identify the relationship(s); analyze the sequence of constructor and destructor invocation; and obtain the output

```
class A
{
    public:
      A() { cout<<”1”; }
     ~A() { cout<<”2”; }
};

class B
{
     A *O1;
     public:
        B() {O1 = new A;}
      ~B() {delete(O1);}
};

class C
{
     A O2;
     public:
       C() { cout<<”5”; }
      ~C() { cout<<”6”; }
};

class D
{
     A *O3;
     public:
       D(A *t) {O3 = t; }
      ~D() { cout<<”6”; }
};

int main() { A *O4 = new A; B O5; C *O6, *O7 = new C; D *O8 = new D(O4); delete(O8); delete(O7); return 0;}
```

Q6. Analyze the sequence of constructor invocation in following C++ program and obtain the output without running the code.

```
class A
{
public:
   A(){cout<<"1";}
};

class B : virtual public A
{
public:
   B() {cout<<" 2";}
};

class C : public B
{
public:
   C(){cout<<" 3";}
};

class D : public C
{
public:
   D(){cout<<"4";}
};

class E
{
public:
   E(){cout<<" 5";}
};

class F : public E
{
public:
   F(){cout<<"6";}
};

class G : virtual public F
{
public:
   G() {cout<<" 7"; }
};

class H : public G
{
public:
   H(){cout<<" 8";}
};

class I : public H
{
public:
   I(){cout<<"9";}
};

class J : public D, public I
{
public:
   J(){cout<<" 10";}
};

int main(){J O; return 0;}
```