LAB - 13

Q1. (Pointers to objects) Define a class Item that is used to store and display the information regarding the item number and its price. Write a program to store and display the details of one items by using both normal object and pointer to object separately. Display appropriate message wherever necessary.

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

class Item
{
private:
    int i, j;

public:
    void
    getdata(){
        cout << "Enter Item number" << endl;
        cin >> i;
        cout << "Enter Item's Price" << endl;
        cin >> j;
    }

    void
    pmdata(void){
        cout << "\nItem's No. " << i << "\n" << "Item's Price: " << j << endl;
    }
};

int main()
{
    Item D1;
    Item *dptr;
    cout << "Using the object" << endl;
    D1.getdata();
    cout << "using the object ";
    D1.pmdata();
    dptr = &D1;
    cout << "using the object pointer ";
    dptr->pmdata();
    return 0;
}
```

```
Using the object
Enter Item number
1
Enter Item's Price
1200
using the object
Item's No. 1
Item's Price: 1200
using the object pointer
Item's No. 1
Item's Price: 1200
```

Q2. Modify the program LE31 by creating an array of n objects using pointers. Show the details of n items by using pointers to object concept.

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

class Item
{
private:
    int i, j;

public:
    void
    getdata(){
        cout << "Enter Item number" << endl;
```

```
        cout << "Enter Item number" << endl;
        cin >> i;
        cout << "Enter Item's Price" << endl;
        cin >> j;
    }

    void
    pmdata(void){
        cout << "\nItem's No. " << i << "\nItem's Price: " << j << endl;
    }
};

int main()
{
    int n, i;
    Item *ob[10];
    cout << "Enter Total no. of Items";
    cin >> n;
    for (i = 0; i < n; i++)
    {
        ob[i]=new Item;
        ob[i]->getdata();
    }
    for (i = 0; i < n; i+
    +){
        ob[i]->pmdata();
    }
    return 0;
}
```

```
Enter Total no. of Items5
Enter Item number
1
Enter Item's Price
1200
Enter Item number
2
Enter Item's Price
120
Enter Item number
3
Enter Item's Price
12
Enter Item number
4
Enter Item's Price
12000
Enter Item number
5
Enter Item's Price
120000

Item's No. 1
Item's Price: 1200

Item's No. 2
Item's Price: 120

Item's No. 3
Item's Price: 12

Item's No. 4
Item's Price: 12000

Item's No. 5
Item's Price: 120000
```

Q3. (Pointers to derived classes) Write a program to illustrate how pointers to a base class is used for both base and derived class.

```
#include <iostream>
using namespace std;
class base
{
public:
    int n1;
    void
    show(){
        cout << "\nn1 = " << n1;
```

```
        cout << "\nn1 = " <<
 n1; }
};
class derive : public base
{
public:
    int n2;
    void
    show(){
        cout << "\nn1 = " << n1;
        cout << "\nn2 = " << n2;
    }
};
int main()
{
    base b;
    base *bptr;
    cout << "Pointer of base class points to it \n";
    bptr = &b;
    bptr->n1 = 50;
    cout << "Access base class via base pointer";
    bptr->show();
    derive d;
    cout << "\n";
    bptr = &d;
    bptr->n1 = 70;
    cout << "Access derive class via base pointer";
    bptr->show();
    return 0;
}
```

```
Pointer of base class points to it
Access base class via base pointer
n1 = 50
Access derive class via base pointer
n1 = 70
```

Q4. (this Pointer) Write a program to display the contains of an object, when local variable's name is same as member's name by using this pointer.

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

class Pointer
{
private:
   int x;

public:
   void setX(int
   x){
      this->x = x;
   }
   void print()
   {
      cout << "x = " << x << endl;
   }
};

int main()
{
   Pointer obj;
   int x = 35;
   obj.setX(x);
   obj.print();
   return 0;
}
```

```
x = 35
```

Q5. (Virtual Function) Define a class ABC. Derive two classes BBC and KBC from ABC. All the classes contains same member function name as display(). The base class pointer always holds the derived class objects.

a) Write a program such that base class pointer or reference will always access/call the base version of
the members available in derived class, do not have any access to the derived class members.
b) Write a program such that base class pointer or reference will always access/call the derived version of
the members available in derived class, do not have any access to the base class members.
Write down the concepts used for bit a) and b) separately.

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

class ABC
{
public:
    virtual void
    display(){
        cout << "\n In Base Class ABC\n";
    }
};
class BBC : public ABC
{
public:
    void
    display(){
        cout << "In Derived Class BBC
 \n"; }
};
class KBC : public ABC
{
public:
    void
    display(){
        cout << "In Derived Class KBC\n";
    }
};
int main()
{
    BBC d1;
    KBC d2;
    ABC *bptr = &d1;
    bptr->display();
    bptr = &d2; bptr-
    >display();
    return 0;
}
```

```
In Derived Class BBC
In Derived Class KBC
```

Q5 b) (Virtual Function) Define a class ABC. Derive two classes BBC and KBC from ABC. All the classes contains same member function name as display(). The base class pointer always holds the derived class objects.
a) Write a program such that base class pointer or reference will always access/call the base version of the
members available in derived class, do not have any access to the derived class members.
b) Write a program such that base class pointer or reference will always access/call the derived version of
the members available in derived class, do not have any access to the base class members.
Write down the concepts used for bit a) and b) separately.

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

class ABC
{
public:
    virtual void display()
    {
```

```
    {
        cout << "\n In Base Class ABC\n";
    }
};
class BBC : public ABC
{
public:
    void
    display(){
        cout << "In Derived Class BBC
 \n"; }
};
class KBC : public ABC
{
public:
    void
    display(){
        cout << "In Derived Class KBC\n";
    }
};
int main()
{
    ABC *ptr = new BBC;
    ptr->display(); ptr =
    new KBC; ptr-
    >display(); return
    0;
}
```

```
In Derived Class BBC
In Derived Class KBC
```

Q6. (Pure Virtual Function) Write a program by modifying LE2.5 b) by making display() as pure virtual function.

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

class ABC
{
public:
    virtual void
    display(){
        cout << "\n In Base Class ABC
 \n"; }
};
class BBC : public ABC
{
public:
    void
    display(){
        cout << "In Derived Class BBC
 \n"; }
};
class KBC : public ABC
{
public:
    void
    display(){
        cout << "In Derived Class KBC\n";
    }
};
int main()
{
    BBC d1;
    KBC d2;
    ABC *bptr = &d1;
    bptr->display();
    bptr = &d2; bptr-
    >display();
    return 0;
}
```

```
In Derived Class BBC
In Derived Class KBC
```