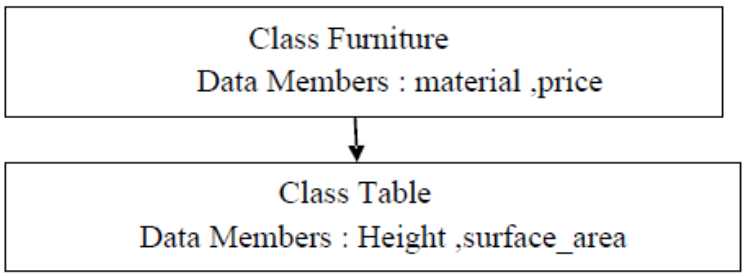
2.B.1 Write a program using function overloading to swap two integer numbers and swap two float numbers.

```
class Overloading
{
public void swap(ref int n, ref int m)
{
int t;
t = n;
n = m;
m = t;
}
public void swap(ref float f1, ref float f2)
{
float f;
f = f1;
f1 = f2;
f2 = f;
}
}
class program
{
static void Main(string[] args)
{
Overloading objOverloading = new Overloading();
int n = 10, m = 20;
objOverloading.swap(ref n, ref m);
Console.WriteLine("N=" + n + "\tM=" + m);
float f1 = 10.5f, f2 = 20.6f;
objOverloading.swap(ref f1, ref f2);
Console.WriteLine("F1=" + f1 + "\tF2=" + f2);
} }
```

2.B.2 Write a program to implement single inheritance



1. Furniture.cs

```
class furniture
  {
    string material;
    float price;
    public void getdata()
    {
       Console.Write("Enter material : ");
       material = Console.ReadLine();
       Console.Write("Enter price : ");
       price = float.Parse(Console.ReadLine());
    }
    public void showdata()
    {
       Console.WriteLine("Material : " + material);
       Console.WriteLine("Price : " + price);
    }
  }
```

2. Table.cs

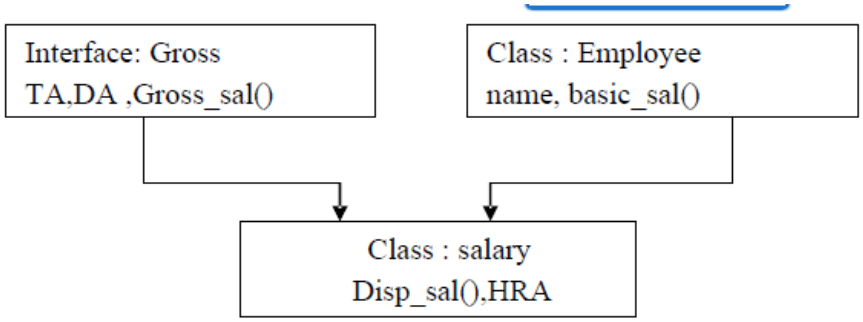
```
  class table:furniture
     {
        int height, surface_area;
     public void getdata()
     {
```

```
        base.getdata();
        Console.Write("Enter height: ");
        height = int.Parse(Console.ReadLine());
        Console.Write("Enter surface area: ");
        surface_area = int.Parse(Console.ReadLine());
        }
        public void showdata()
        {
        base.showdata();
        Console.WriteLine("Height : " + height);
        Console.WriteLine("Surface Area : " + surface_area);
    }
        }
```

3. Program.cs

```
class Program
  {
    static void Main(string[] args)
    {
       table t1 = new table();
       t1.getdata();
       t1.showdata();
       Console.ReadKey();
    }
  }
```

2.B.2 Program to implement the multiple inheritance using interface.



1. Gross.cs

```
interface Gross
  {
    int ta
    {
      get;
      set;
    }
    int da
    {
      get;
      set;
    }
    int GrossSal();
  }
```

2. Employee.cs

```
class Employee
  {
    string name;
    public Employee(string name)
    { this.name = name; }
    public int BasicSal(int basicSal)
```

```
    { return basicSal; }
    public void ShowData()
    {
        Console.WriteLine("Name : " + name);
    }
}
```

3. Salary.cs

```
class Salary:Employee,Gross
{
    int hra;
    public Salary(string name, int hra)
        : base(name)
    { this.hra = hra; }
    public int ta
    {
        get { return S_ta; }
        set { S_ta = value; }
    }
    private int S_ta;
    public int da
    {
        get { return S_da; }
        set { S_da = value; }
    }
    private int S_da;
    public int GrossSal()
    {
        int gSal;
        gSal = hra + ta + da + BasicSal(15000);
        return gSal;
```

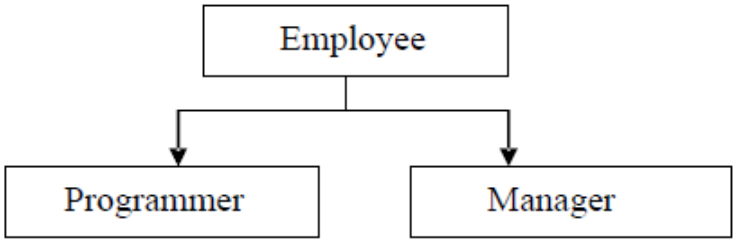
```
    }
    public void dispSal()
    {
        base.ShowData();
        Console.WriteLine("Gross Sal : " + GrossSal());
    }
}
```

4. Program.cs

```
class Program
{
    static void Main(string[] args)
    {
        Salary s = new Salary("Mac", 35000);
        s.da = 20000;
        s.ta = 30000;
        s.dispSal();
        Console.ReadKey();

    }
}
```

2.B.2 Write a program for class hierarchy for the Employee where the base class is Employee and derived class and Programmer and Manager.



1. Programmer.cs

```
public void display()
    {
       Console.WriteLine(" Display of Programmer class called ");
    }
```

2. Manager.cs

```
public void display()
    {
       Console.WriteLine("Display of manager class called ");
    }
```

3. Employee.cs

```
public virtual void display()
    {
       Console.WriteLine("Display of employee class called ");
    }
```

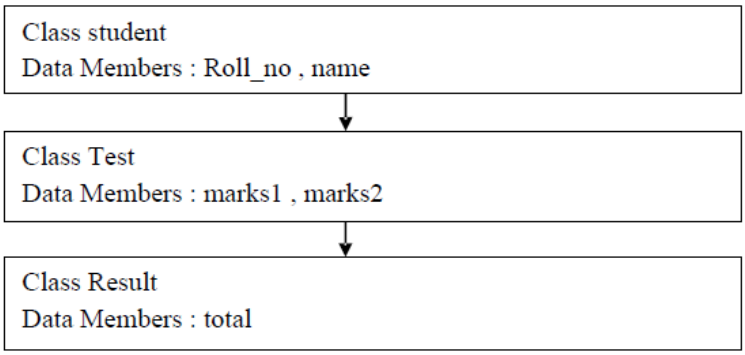
4. Program.cs

```
class Program
  {
     static void Main(string[] args)
     {
        Programmer objProgrammer;
Manager objManager;
```

```
Employee objEmployee;
Console.WriteLine("Whose details you want to use to see \n 1.Programmer \n 2.Manager");
int choice=int.Parse(Console.ReadLine());
if(choice==1)
{
objProgrammer=new Programmer();
objProgrammer.display();
}
else if(choice==2)
{
objManager=new Manager();
objManager.display();
}
else if (choice == 3)
{
   objEmployee = new Employee();
   objEmployee.display();
}
else
{
Console.WriteLine("Wrong choice entered");

}
Console.ReadKey();
      }
   }
```

2.B.2 Write a program to implement multilevel inheritance



1. student.cs

```
class student
   {
      int roll_no;
      string name;
      public student(int roll_no, string name)
      {
         this.roll_no = roll_no;
         this.name = name;
      }
      public student() { }
      public void display()
      {
         Console.WriteLine("Roll no: " + roll_no);
         Console.WriteLine("Name: " + name);
      }
   }
```

2. Test.cs

```
class Test:student
   {
```

```
    int marks1, marks2;
public Test(int roll_no, string name, int marks1, int marks2)
: base(roll_no, name)
{
this.marks1 = marks1;
this.marks2 = marks2;
}
public int getMarks1()
{
   return marks1;
}
public int getMarks2()
{
   return marks2;
}
public void dispaly()
{
   base.display();
   Console.WriteLine("Marks1: " + marks1);
   Console.WriteLine("Marks2: " + marks2);
}
   }
```

3. Result.cs

```
class Result:Test
   {
      int total;
      public Result(int roll_no, string name, int marks1, int marks2)
```

```
        : base(roll_no, name, marks1, marks2)
    {
        total = getMarks1() + getMarks2();
    }
    public void display()
    {
        base.display();
        Console.WriteLine("Total: " + total);
    }
}
```

4. Program.cs

```
class Program
{
    static void Main(string[] args)
    {
        Result r1 = new Result(101, "Prachit", 50, 70);
        r1.display();
        Console.ReadKey();
    }
}
```

2.B.3 Constructor overloading

```
class ADD
  {

    int x, y;
    double f;
    string s;

    // 1st constructor
    public ADD(int a, double b)
    {
       x = a;
       f = b;
    }

    // 2nd constructor
    public ADD(int a, string b)
    {
       y = a;
       s = b;
    }

    // showing 1st constructor's result
    public void show()
    {
       Console.WriteLine("1st constructor (int + double): {0} ",
                                          (x + f));
    }
```

```
    // shows 2nd constructor's result
    public void show1()
    {
        Console.WriteLine("2nd constructor (int + string): {0}",
                                            (s + y));
    }
}
class Program
{
    static void Main(string[] args)
    {
        ADD g = new ADD(10, 20.2);

        // calling the method
        g.show();



        // Creating instance and
        // passing arguments
        // It will call the second constructor
        ADD q = new ADD(10, "Roll No. is ");

        // calling the method
        q.show1();
        Console.ReadKey();
    }
}
```