Unit 1 :

3. Temperature converter

```
//to convert temperature from Fahrenheit to Celsius
// F = 9*C/5 + 32
// C = (F-32)*5/9

class Temperature
{
   public static void Main()
   {
      float ans;
      Console.Write("Enter temperature : ");
      float temp = Convert.ToInt32(Console.ReadLine());
      Console.WriteLine("1. Convert from Celsius to Fahrenheit");
      Console.WriteLine("2. Convert from Fahrenheit to Celsius");
      Console.Write("Enter your choice : ");
      int choice = Convert.ToInt32(Console.ReadLine());
      if (choice == 1)
      {
         ans = (((9 * temp) / 5) + 32);
         Console.WriteLine("Temperature in Fahrenheit = {0}", ans);
      }
      else if (choice == 2)
      {
         ans = (((temp - 32) * 5) / 9);
         Console.WriteLine("Temperature in Celsius = {0}", ans);
      }
      else
         Console.WriteLine("Enter a valid choice");
   }
}
```

4. Prime Number

```
//to check whether given number is prime or not

class PrimeCheck
{
   public static void Main()
   {
      int num;
      Boolean flag = true;
      string val;
      Console.Write("Enter number : ");
      val = Console.ReadLine();
      num = Convert.ToInt32(val);
      for (int i = 2; i < num / 2 && flag == true; i++)
      {
         if (num % i == 0)
            flag = false;
      }
      if (num == 1)
         Console.WriteLine("{0} is neither prime nor composite", num);
      else if (flag)
         Console.WriteLine("{0} is a prime number", num);
      else
         Console.WriteLine("{0} is a composite number", num);
   }
```

```
}
```

6. Vowels

```
namespace Vowels
{
    internal class Program
    {
        static void Main(string[] args)
        {
            Console.Write("Enter a string : ");
            string s = Console.ReadLine();
            char [] n = s.ToCharArray();
            int count_a = 0;
            int count_e = 0;
            int count_i = 0;
            int count_o = 0;
            int count_u = 0;

            foreach(char c in n)
            {
                switch(c)
                {
                    case 'a':
                    case 'A':
                        count_a++;
                        break;
                    case 'e':
                    case 'E':
                        count_e++;
                        break;
                    case 'i':
                    case 'I':
                        count_i++;
                        break;
                    case 'o':
                    case 'O':
                        count_o++;
                        break;
                    case 'u':
                    case 'U':
                        count_u++;
                        break;
                }
            }

            int total = count_a + count_e+ count_i + count_o + count_u;

            Console.WriteLine("Count of a = {0}", count_a);
            Console.WriteLine("Count of e = {0}", count_e);
            Console.WriteLine("Count of i = {0}", count_i);
            Console.WriteLine("Count of o = {0}", count_o);
            Console.WriteLine("Count of u = {0}", count_u);
            Console.WriteLine("Totel vowels = {0}", total);
        }
    }
}
```

8. Fibonacci

```
namespace Fibonacci
{
    internal class Program
    {
        static void Main(string[] args)
        {
            Console.Write("Enter the number of terms : ");
            int n = Convert.ToInt32(Console.ReadLine());
            int a = 1;
            int b = 1;
            int sum;
            int count = 2;
            Console.Write("{0} \t {1} \t ", a, b);
            while(count < n)
            {
                sum = a + b;
                Console.Write("{0} \t ", sum);
                count++;
                a = b;
                b = sum;
            }
        }
    }
}
```

9. Factorial

```
namespace Factorial
{
    internal class Program
    {
        static void Main(string[] args)
        {
            Console.Write("Enter number : ");
            int n = Convert.ToInt32(Console.ReadLine());
            int prod = 0;
            int sum = 1;
            for(int i=1; i<=n; i++)
            {
                sum *= i;
            }
            Console.WriteLine("Factorial = {0}",sum);
        }
    }
}
```

Unit 2 :

1. Parameterized Constructor

```
namespace Multiplication_with_Parameterised_Constructor
{
   class Solution
   {
      public int a, b, prod;

      public Solution(int a,int b)
      {
         this.a = a;
         this.b = b;
         prod = a * b;
      }
   }

   class Program
   {
      static void Main(string[] args)
      {
         Console.Write("Enter a : ");
         int a = Convert.ToInt32(Console.ReadLine());
         Console.Write("Enter b : ");
         int b = Convert.ToInt32(Console.ReadLine());
         Solution s = new Solution(a, b);
         Console.WriteLine("a * b = {0}",s.prod);
      }
   }
}
```

2. Operator Overloading

```
namespace Operator_Overloading
{
   public class Test
   {
      public int a;

      public Test(int a)
      {
         this.a = a;
      }

      public Test() { }

      public static Test operator +(Test t1, Test t2)
      {
         Test t = new Test();
         t.a = t1.a + t2.a;
         return t;
      }
   }

   public class Program
   {
      static void Main(string[] args)
      {
```

```
            Test t1 = new Test(10);
            Test t2 = new Test(20);

            Test t3 = t1 + t2;

            Console.WriteLine("t1 + t2 = {0}", t3.a);
        }
    }
}
```

3. Function Overloading

```
namespace Function_Overloading
{
    class Overload
    {
        public int add(int a, int b)
        {
            return a + b;
        }

        public float add(float a, float b)
        {
            return a + b;
        }

        public string add(string a, string b)
        {
            return a + b;
        }
    }
    class Program
    {
        static void Main(string[] args)
        {
            Overload ov = new Overload();
            Console.WriteLine("{0}", ov.add(10, 20));
            Console.WriteLine("{0}", ov.add(10.5f, 20.7f));
            Console.WriteLine(ov.add("Method", "Overloading"));
        }
    }
}
```

4. Multiple Inheritance

```
namespace Multiple_Inheritance
{
    public class Parent1
    {
        Parent1()
        {
            Console.WriteLine("Parent Class 1 initialised.");
        }
    }

    public class Parent2
    {
        Parent2()
        {
```

```
            Console.WriteLine("Parent Class 2 initialised.");
        }
    }

    public class Child : Parent1 , Parent2
    {
        Child()
        {
            Console.WriteLine("Child of Parent1 and Parent2 initialised.");
        }
    }

    class Program
    {
        public static void Main(string[] args)
        {
            Console.WriteLine("Implementing multiple inheritance");
            Child c = new Child();
        }
    }
}
```

5. Multilevel Inheritance

```
namespace Multilevel_Inheritance
{
    public class Parent
    {
        public Parent()
        {
            Console.WriteLine("Parent Class initialised.");
        }
    }

    public class Child : Parent
    {
        public Child()
        {
            Console.WriteLine("Child Class initialised.");
        }
    }

    public class GrandChild : Child
    {
        public GrandChild()
        {
            Console.WriteLine("GrandChild Class initialised.");
        }
    }

    class Program
    {
        public static void Main(string[] args)
        {
            Console.WriteLine("Implementing Multilevel Inheritance.");
            GrandChild gc = new GrandChild();
        }
    }
}
```

6. Hybrid Inheritance

```
namespace Hybrid_Inheritance
{
    public class GrandParent
    {
        public GrandParent()
        {
            Console.WriteLine("GrandParent Class initialised.");
        }
    }

    public class Parent : GrandParent
    {
        public Parent()
        {
            Console.WriteLine("Parent Class initialised.");
        }
    }

    public class Child1 : Parent
    {
        public Child1()
        {
            Console.WriteLine("Child Class 1 initialised.");
        }
    }

    public class Child2 : Parent
    {
        public Child2()
        {
            Console.WriteLine("Child Class 2 initialised");
        }
    }

    class Program
    {
        public static void Main(string[] args)
        {
            Console.WriteLine("This is what hybrid inheritance looks like : ");
            Console.WriteLine("                    ---> Child1        ");
            Console.WriteLine("                    |                  ");
            Console.WriteLine("GrandParent ---> Parent |");
            Console.WriteLine("                    |                  ");
            Console.WriteLine("                    ---> Child2        ");
            Console.WriteLine("Implementing Hybrid Inheritance \n");
            Child1 c1 = new Child1();
            Console.WriteLine();
            Child2 c2 = new Child2();
        }
    }
}
```