C# and .NET LABORATORY

(**Semester -VI** of B.Tech)

As per the curricullam and syllabus
of
Bharath Institute of Higher Education & Research

(C# and .NET Lab manual)

Bharath
INSTITUTE OF HIGHER EDUCATION AND RESEARCH
(Declared as Deemed - to - be - University under section 3 of UGC Act 1956)
ACCREDITED WITH 'A' GRADE BY NAAC

NEW EDITION

PREPARED BY

DR. M.K.VIDHYALAKSHMI

Bharath

INSTITUTE OF HIGHER EDUCATION AND RESEARCH

(Declared as Deemed-to-be University under section 3 of UGC Act, 1956)

(Vide Notification No. F.9-5/2000 - U.3, Ministry of Human Resource Development, Govt. of India, dated 4th July 2002)

SCHOOL OF COMPUTING

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

LAB MANUAL

SUBJECT NAME: Data Warehousing and Data Mining Laboratory

SUBJECT CODE: BCS6L2

Regulation R2015
(2015-2016)

BCS6L2	C# AND .NET LABORATORY	L	T	P	C
	Total Contact Hours - 30	0	0	3	2
	Prerequisite –Object Oriented Programming using C++,Java Programming				
	Lab Manual Designed by – Dept. of Computer Science and Engineering.				

OBJECTIVES

The main Objective of this course is student know about windows, Web and Console Applications.

COURSE OUTCOMES (COs)

CO1	Display proficiency in C# by building stand-alone applications in the .NET framework using C#.
CO2	Create distributed data-driven applications using the .NET Framework, C#, SQL Server and ADO.NET
CO3	Create web-based distributed applications using C#, ASP.NET, SQL Server and ADO.NET
CO4	Utilize DirectX libraries in the .NET environment to implement 2D and 3D animations and game-related graphic displays and audio.
CO5	Utilize XML in the .NET environment to create Web Service-based applications and components.
CO6	Understand the concept of Web Applications.

MAPPING BETWEEN COURSE OUTCOMES & PROGRAM OUTCOMES
(3/2/1 INDICATES STRENGTH OF CORRELATION) 3- High, 2- Medium, 1-Low

COs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
CO1		3	3		3			3			3		2	3	
CO2		3	3		3			3			3		3	3	
CO3		3	3		3			3			3		3	3	
CO4		3	3		3			3			3		3	3	
CO5		2									2		2	3	
CO6			3		2			2			3		2	3	

Category	Professional Core (PC)
Approval	37th Meeting of Academic Council, May 2015

LIST OF EXPERIMENTS

1. Classes and objects
2. Inheritance
3. Operator overloading
4. Threading
5. Events and delegates
6. Working with windows forms controls
7. Validating data
8. Creating custom dialog box
9. Designing an MDI application with menu
10. Retrieving data from a SQL database
11. Manipulating data in a connected environment
12. Manipulating data in a disconnected environment

DATA WAREHOUSING AND DATA MINING LABORATORY- BCS6L1

LIST OF EXPERIMENTS

	NAME OF THE EXPERIMENT
1	Basic c# programs
2	Classes and objects
3	Inheritance
4	Operator overloading
5	Threading
6	Events and delegates
7	Working with windows forms controls
8	Validating data
9	Creating custom dialog box and Designing an MDI application with menu
10	Retrieving Data From Database & Working With Disconnected Environment

CONTENT

EXPERIMENT 1 – BASIC C# PROGRAMS

AIM

To understand about basics of C# and execute simple c# programs to perform the following actions:

(a) Calculate Hypotenuse of triangle using dynamic initialization of variables

(b) To get input from the user and perform calculations

(c) Calculate the quadrant for the coordinates using if..else…ladder

(d) Check whether the alphabet is a vowel or not using switch..case…

(e) To understand about for..each loop and strings

ALGORITHM

Step 1: Open Visual Studio Express edition 2010
Step 2: Click File→New project→Select C# under installed tab and select console application
Step 3: Give name for your application and click OK
Step 4: Give any class name and declare variables and write methods
Step 5: Create objects for classes to execute methods
Step 6: Click save and click run button for execution

PROGRAMS:

(a) Hypotenuse of triangle:

```
class DynInit
 {
static void Main()
{
// Length of sides.
double s1 = 4.0;
double s2 = 5.0;
// dynamically initialize hypotenuse
double hypot = Math.Sqrt( (s1 * s1) + (s2 * s2) );
Console.Write("Hypotenuse of triangle with sides " + s1 + " by " + s2 + " is ");
Console.WriteLine("{0:#.###}.", hypot); //format to display using 3 decimal values
Console.Readkey();
}
}
```

(b) To get input from the user and perform calculations

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

```
            Console.Write("Enter a number: ");
            int num1 = Convert.ToInt32(Console.ReadLine());

            Console.Write("Enter another number: ");
            int num2 = Convert.ToInt32(Console.ReadLine());

            Console.WriteLine("{0} + {1} = {2}", num1, num2, num1 + num2);
            Console.WriteLine("{0} - {1} = {2}", num1, num2, num1 - num2);
            Console.WriteLine("{0} x {1} = {2}", num1, num2, num1 * num2);
            Console.WriteLine("{0} / {1} = {2}", num1, num2, num1 / num2);
            Console.WriteLine("{0} mod {1} = {2}", num1, num2, num1 % num2);
            Console.ReadKey();
    }
```

(c) Calculate the quadrant for the coordinates using if..else ladder

```
    class Program
        {
            static void Main(string[] args)
            {
                int co1, co2;

                Console.Write("\n\n");
                Console.Write("Find the quadrant in which the coordinate point lies:\n");
                Console.Write("-----------------------------------------------------");
                Console.Write("\n\n");

                Console.Write("Input the value for X coordinate :");
                co1 = Convert.ToInt32(Console.ReadLine());
                Console.Write("Input the value for Y coordinate :");
                co2 = Convert.ToInt32(Console.ReadLine());

                if (co1 > 0 && co2 > 0)
                    Console.Write("The coordinate point ({0} {1}) lies in the First quandrant.\n\n",
                                                                                    co1, co2);
              else if (co1 < 0 && co2 > 0)
                    Console.Write("The coordinate point ({0} {1}) lies in the Second
                                                                    quandrant.\n\n", co1, co2);
             else if (co1 < 0 && co2 < 0)
                    Console.Write("The coordinate point ({0} {1}) lies in the Third
                                                                     quandrant.\n\n", co1, co2);
            else if (co1 > 0 && co2 < 0)
                    Console.Write("The coordinate point ({0} {1}) lies in the Fourth
                                                                     quandrant.\n\n", co1, co2);
           else if (co1 == 0 && co2 == 0)
                   Console.Write("The coordinate point ({0} {1}) lies at the origin.\n\n", co1, co2);

            Console.ReadKey();
        }
```

(d) Check whether the alphabet is a vowel or not using switch..case…

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

```
        {
            char ch;
            Console.Write("\n\n");
            Console.Write("check whether the input alphabet is a vowel or not:\n");
            Console.Write("-----------------------------------------------------");
            Console.Write("\n\n");

            Console.Write("Input an Alphabet (A-Z or a-z) : ");
            ch = Convert.ToChar(Console.ReadLine().ToLower());
            int i = ch;
            if (i >= 48 && i <= 57)
            {
               Console.Write("You entered a number, Please enter an alphabet.");
            }
            else
            {
               switch (ch)
               {
                  case 'a':
                     Console.WriteLine("The Alphabet is vowel");
                     break;
                  case 'i':
                     Console.WriteLine("The Alphabet is vowel");
                     break;
                  case 'o':
                     Console.WriteLine("The Alphabet is vowel");
                     break;
                  case 'u':
                     Console.WriteLine("The Alphabet is vowel");
                     break;
                  case 'e':
                     Console.WriteLine("The Alphabet is vowel");
                     break;
                  default:
                     Console.WriteLine("The Alphabet is a consonant");
                     break;
               }
            }
           Console.ReadKey();
      }
```

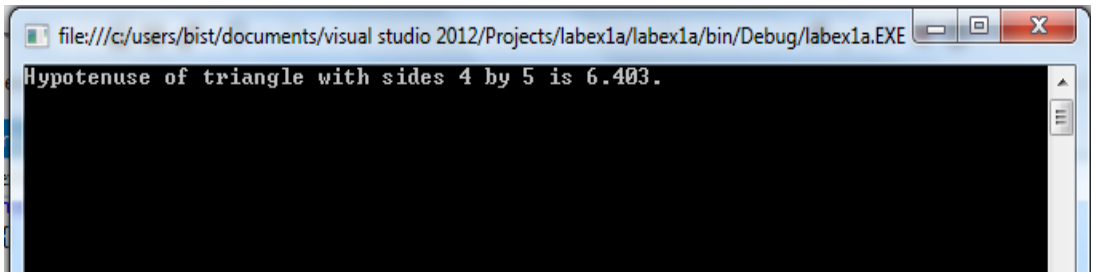
(e) To understand about for..each loop and strings

```
  //print the length of the string without using library functions
   class Program
     {
     static void Main(string[] args)
     {
        string str; /* Declares a string of size 100 */
         int length= 0;
         Console.Write("\n\nFind the length of a string :\n");
          Console.Write("---------------------------------\n");
         Console.Write("Input the string : ");
         str = Console.ReadLine();
```

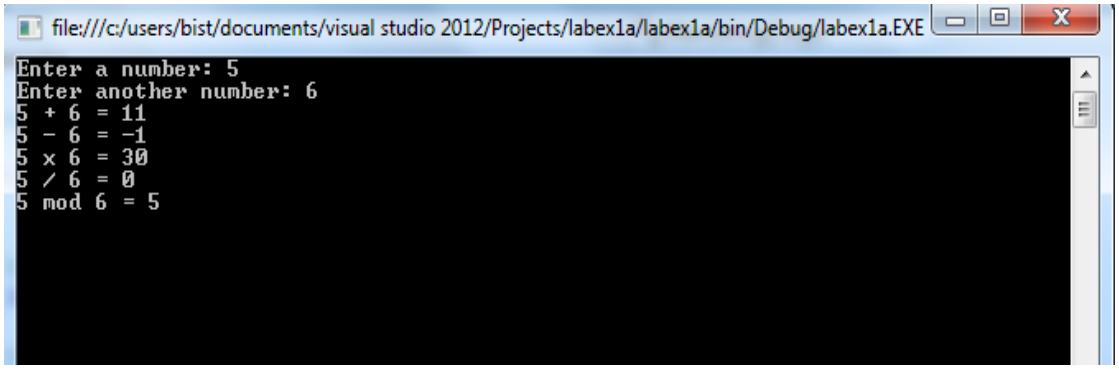
```
        foreach(char chr in str)
        {
            length+= 1;
        }
  Console.Write("Length of the string is : {0}\n\n", length);
  Console.ReadKey();
}
```

OUTPUT

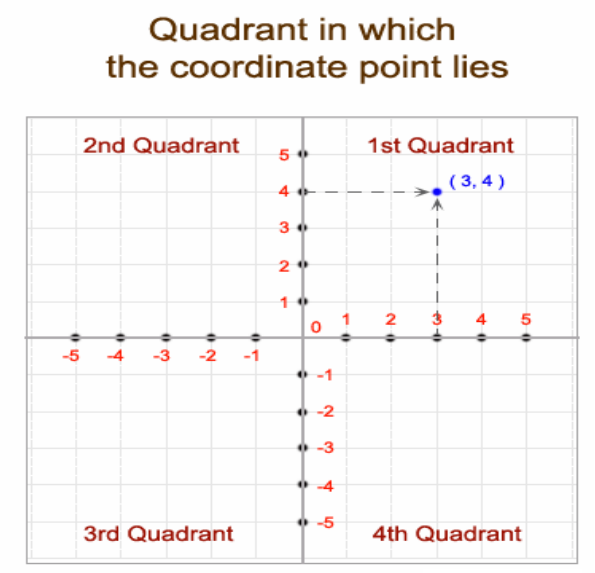
(a) Hypotenuse

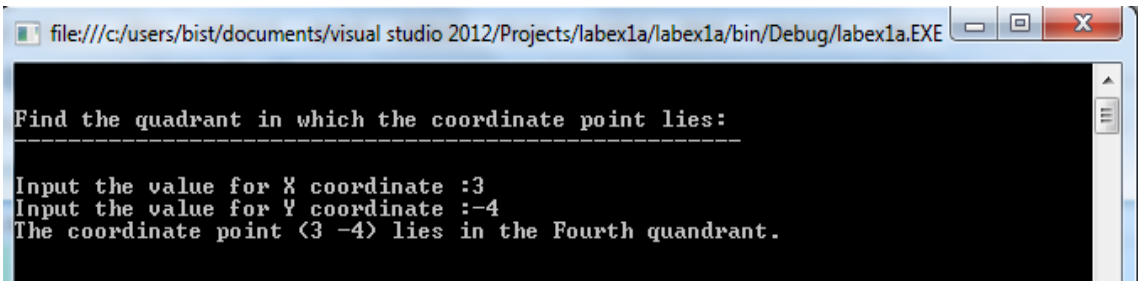


(b) Get input and perform calculations

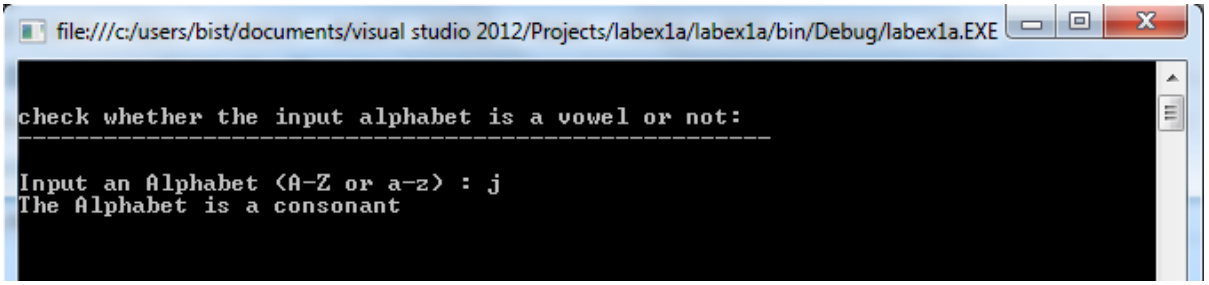


(c) Coordinates and Quadrant

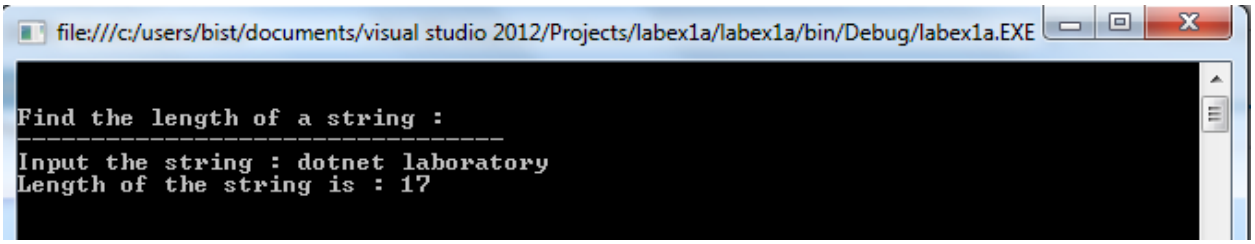




(c) Check the alphabet



(e) Strings & foreach loop



Result:

Thus the program in C# programs is executed successfully and the output is verified

EXPERIMENT 2 – CLASSES & OBJECTS

AIM

To develop a C# application to print the students list using classes and objects

ALGORITHM

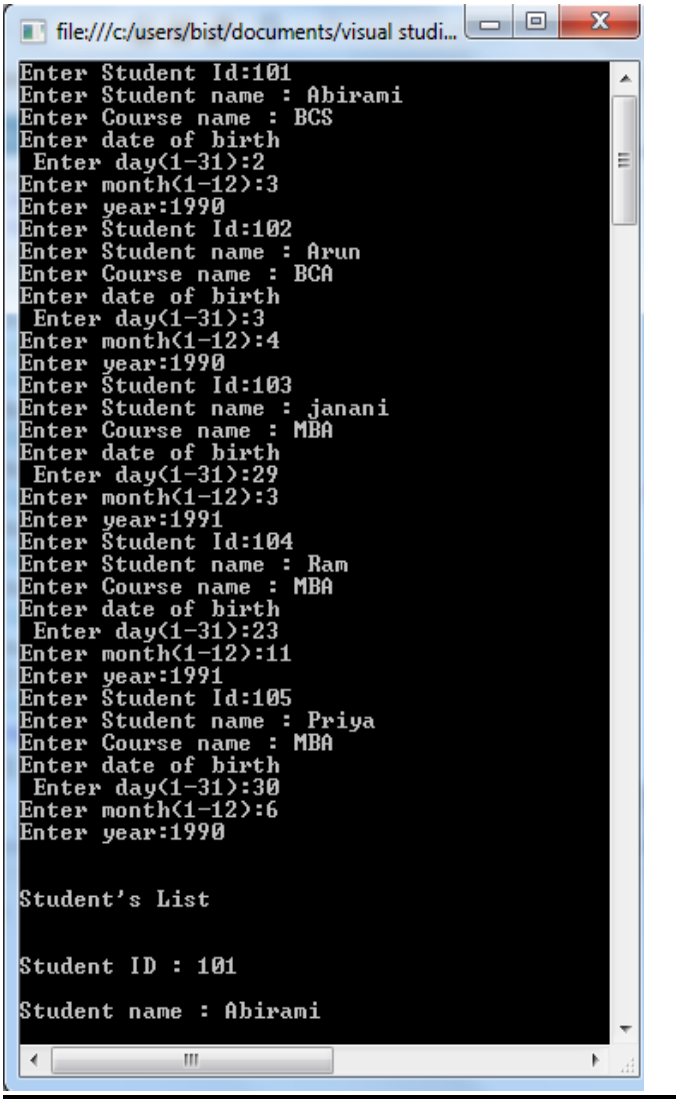
1) Open Visual Studio Express 2010
2) Create a new project → select visual c# →console application → give any name → Ok
3) Create a class and declare necessary array variables
4) Create an object for the class and using for loop iterate it and get inout from the user to feed student details
5) Display the student details
6) Save and execute the program

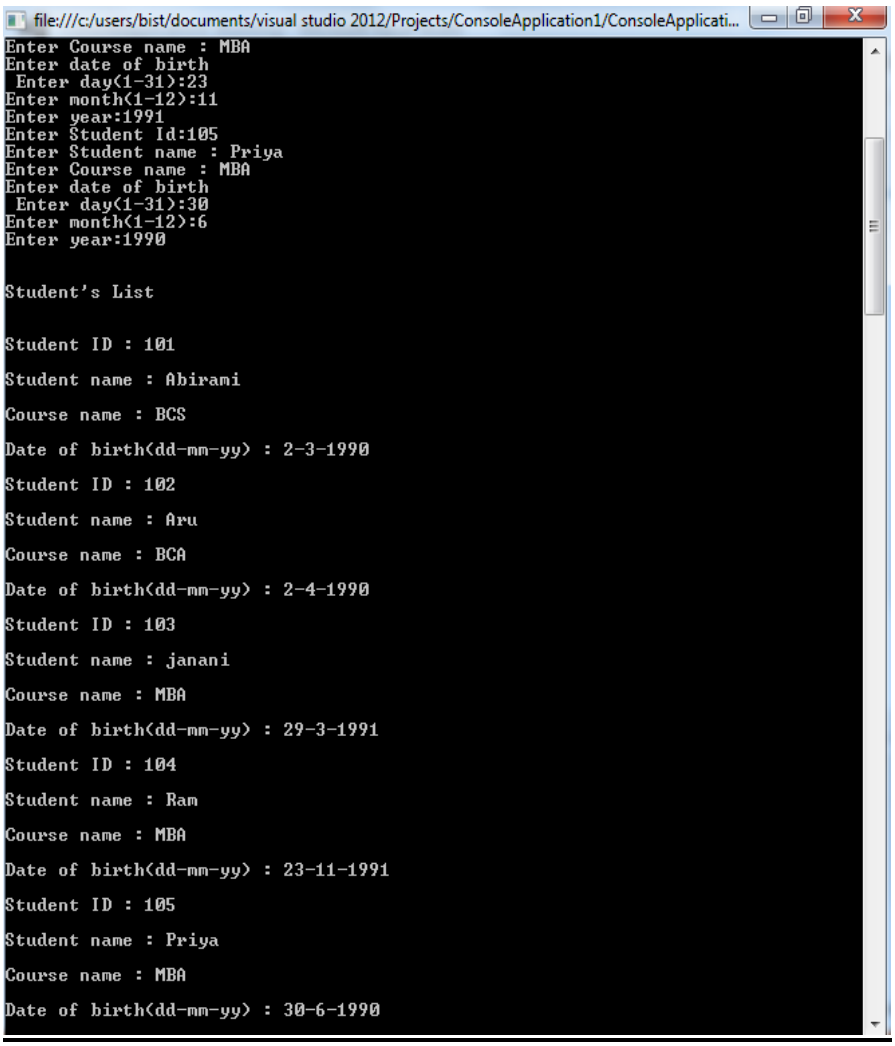
PROGRAM

```
using System;
namespace ConsoleApplication1
{
   class Student
   {
    public  int[] studid = new int[5];
     public int[] day = new int[5];
     public int[] month = new int[5];
     public int[] year = new int[5];
     public string[] name = new string[5];
     public string[] cname = new string[5];
      public void details()
   {
      Console.WriteLine("Implementation of Classes and Objects ");
      Console.WriteLine("**************************************");
      Console.WriteLine("Enter students details and you can view those details");
      Console.WriteLine("-----------------------------------------------------");
   }
        class Ex2
       {
         static void Main(string[] args)
         {
           Student s = new Student();
          s.details();
           int i;
           for (i = 0; i < 5; i++)
           {
           Console.Write("Enter Student Id:");
           s.studid[i] = Convert.ToInt32(Console.ReadLine());
          Console.Write("Enter Student name : ");
           s.name[i] = Console.ReadLine();
```

```
            Console.Write("Enter Course name : ");
            s.cname[i] = Console.ReadLine();
            Console.Write("Enter date of birth\n Enter day(1-31):");
            s.day[i] = Convert.ToInt32(Console.ReadLine());
            Console.Write("Enter month(1-12):");
            s.month[i] = Convert.ToInt32(Console.ReadLine());
             Console.Write("Enter year:");
                  s.year[i] = Convert.ToInt32(Console.ReadLine());
        }
     Console.WriteLine("\n\nStudent's List\n");
    for (i = 0; i < 5; i++)
    {
       Console.WriteLine("\nStudent ID : " + s.studid[i]);
       Console.WriteLine("\nStudent name : " + s.name[i]);
       Console.WriteLine("\nCourse name : " + s.cname[i]);
       Console.WriteLine("\nDate of birth(dd-mm-yy) : " + s.day[i] + "-" + s.month[i] + "-" +
        s.year[i]);
       Console.ReadKey();
    }
    }
    }
}
```

Output:


file:///c:/users/bist/documents/visual studi...
Enter Student Id:101
Enter Student name : Abirami
Enter Course name : BCS
Enter date of birth
Enter day(1-31):2
Enter month(1-12):3
Enter year:1990
Enter Student Id:102
Enter Student name : Arun
Enter Course name : BCA
Enter date of birth
Enter day(1-31):3
Enter month(1-12):4
Enter year:1990
Enter Student Id:103
Enter Student name : janani
Enter Course name : MBA
Enter date of birth
Enter day(1-31):29
Enter month(1-12):3
Enter year:1991
Enter Student Id:104
Enter Student name : Ram
Enter Course name : MBA
Enter date of birth
Enter day(1-31):23
Enter month(1-12):11
Enter year:1991
Enter Student Id:105
Enter Student name : Priya
Enter Course name : MBA
Enter date of birth
Enter day(1-31):30
Enter month(1-12):6
Enter year:1990
Student's List
Student ID : 101
Student name : Abirami



Result:

Thus the program in C# for classes and objects is executed successfully and the output is verified

EXPERIMENT 3(a) – INHERITANCE

AIM

To develop a C# application to implement inheritance concepts

(a) **Single Inheritance**

(b) **Multilevel Inheritance**

(c) **Multiple Inheritance**

ALGORITHM (a)

1. Open Visual Studio Express 2010
2. Create a new project → select visual c# →console application → give any name → Ok
3. Create a class name Rectangle and declare necessary variables and methods
4. Create a subclass named Tabletop to inherit Rectangle class
5. Create another class named Execute Rectangle and create objects for derived class and execute the methods
6. Save and execute the program

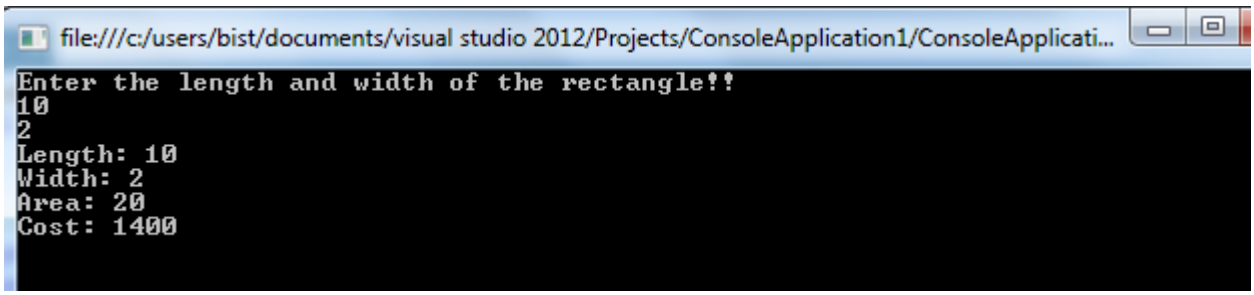
PROGRAM

(a) **Single Inheritance**

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace ConsoleApplication1
{
   class Rectangle
   {
      protected double length,width;

      public void getdata()
      {
         Console.WriteLine("Enter the length and width of the rectangle!!");
         length = Convert.ToDouble(Console.ReadLine());
         width = Convert.ToDouble(Console.ReadLine());
      }
      public double GetArea()
      {
         return length * width;
      }
      public void Display()
      {
         Console.WriteLine("Length: {0}", length);
         Console.WriteLine("Width: {0}", width);
         Console.WriteLine("Area: {0}", GetArea());
      }
   }//end class Rectangle
```

```
    class Tabletop : Rectangle
    {
        private double cost;


        public double GetCost()
        {
            double cost;
            cost = GetArea() * 70;
            return cost;
        }
        public void Display()
        {
            base.Display();
            Console.WriteLine("Cost: {0}", GetCost());
        }
    }
    class ExecuteRectangle
    {
        static void Main(string[] args)
        {
            Tabletop t = new Tabletop();
            t.getdata();
            t.Display();
            Console.ReadKey();
        }
    }

}
```

Output:



Ex-3-B MULTILEVEL INHERITANCE

ALGORITHM (b)

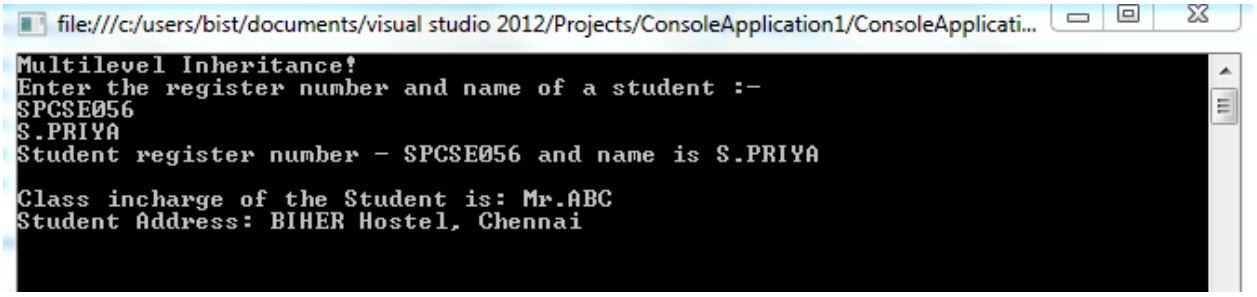
1) Open Visual Studio Express 2010
2) Create a new project → select visual c# →console application → give any name → Ok
3) Create a class name "Person" and declare necessary variables and methods
4) Create a subclass named "Student" to inherit Person class
5) Create another subclass named "Details" to inherit from Student
6) Create a class named "TestClass" and create objects for derived class and execute the methods
7) Save and execute the program

PROGRAM:

```
namespace ConsoleApplication1
{
   public class Person
   {
      protected string regno, name;
      public void get()
      {
         Console.WriteLine("Multilevel Inheritance!");
         Console.WriteLine("Enter the register number and name of a student :-");
         regno = Console.ReadLine();
         name = Console.ReadLine();
      }

      public virtual void display()
      {

         Console.WriteLine("Student register number - {0} and name is {1}", regno, name);
         Console.ReadLine();
      }

   }
   class Student : Person
   {
      public string Classincharge = "Mr.ABC";
      public override void display()
      {
         base.display();
         Console.WriteLine("Class incharge of the Student is: {0}", Classincharge);
      }
   }
   class Details : Student
   {
      private string StudentAddress = "BIHER Hostel, Chennai";
      public void display()
      {
         Console.WriteLine("Student Address: {0}", StudentAddress);
```

```
    }
}
class TestClass
{
    public static void Main()
    {
        Student s = new Student();
        s.get();
        s.display();
        Details d = new Details();
        d.display();
        Console.ReadKey();
    }}}
```

OUTPUT:



Ex-3-C MULTIPLE INHERITANCE

ALGORITHM (c)

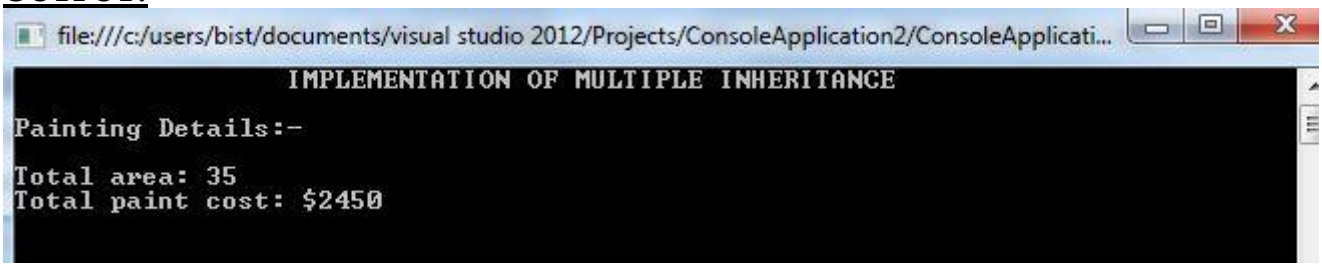
1) Open Visual Studio Express 2010
2) Create a new project → select visual c# →console application → give any name → Ok
3) Create a class named "Shape" and declare necessary variables and methods
4) Create an interface named "PaintCost" to declare methods for classes
5) Create a subclass named "Rectangle" to inherit from Shape class and interface
6) Create a class named "Test" and create objects for derived class and execute the methods
7) Save and execute the program

PROGRAM:

```
namespace ConsoleApplication2
{
   class Shape
   {
      protected int width,height;
      public void setWidth(int w)
      {
         width = w;
      }
      public void setHeight(int h)
      {
         height = h;
      }

   }

    public interface PaintCost    / /interface declarations
   {
      int getCost(int area);
   }

   // Derived class inherits shape class & interface
   class Rectangle : Shape, PaintCost
   {
      public int getArea()
      {
         return (width * height);
      }
      public int getCost(int area)
      {
         return area * 70;
      }
   }
   class Test
   {
```

```
        static void Main(string[] args)
        {
            Rectangle Rect = new Rectangle();
            int area;
            Rect.setWidth(5);
            Rect.setHeight(7);
            area = Rect.getArea();

            Console.WriteLine("\t \t IMPLEMENTATION OF MULTIPLE INHERITANCE \n");
            Console.WriteLine("Painting Details:- \n");

            Console.WriteLine("Total area: {0}", Rect.getArea());
            Console.WriteLine("Total paint cost: ${0}", Rect.getCost(area));
            Console.ReadKey();
        }
    }
}
```

OUTPUT:



Result:

Thus the program in C# programs inheritance concepts is implemented, executed successfully and the output is verified

EXPERIMENT 4.a – OPERATOR OVERLOADING

AIM

To develop a console application to implement operator overloading concept in C#

(a) Unary Operator Overloading

(b) Binary Operator Overloading

a) **UNARY OPERATOR OVERLOADING**

ALGORITHM

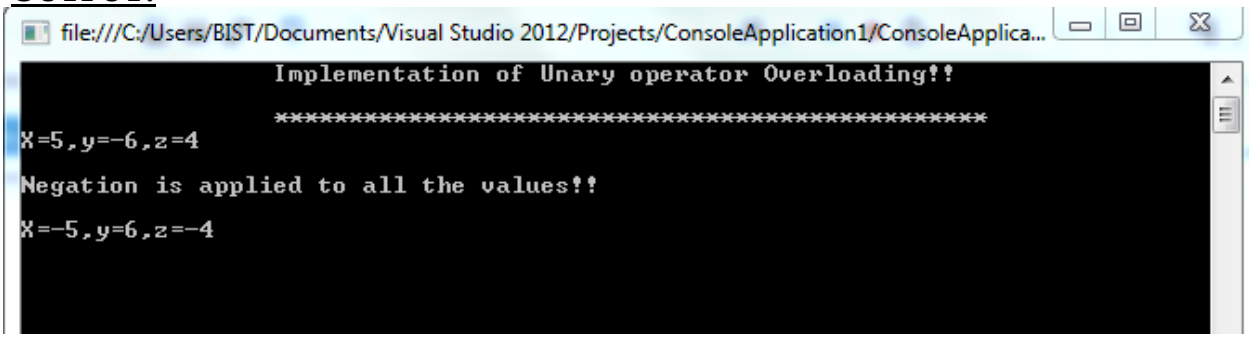
Step 1: Create a new project and a console application in Visual Studio 2012
Step 2: Create a class named "Negate" and declare three variables
Step 3: Write a method to overload minus operator to negate the values given through main method
Step 4: Create objects for negate classes and so that unary minus operator when applied to an object will change the sign of each of its data items
Step 5: Display the values that are negated

PROGRAM:

```
namespace ConsoleApplication6
{
   class Negate
   {
      int x, y, z;
      public Negate (int a, int b, int c)
      {
         x = a;
         y = b;
         z = c;
      }
      void display()
      {
        Console.WriteLine("X={0},y={1},z={2}", x, y, z);
      }

      public static Negate operator -( Negate c)
       {
           c.x = -c.x;
           c.y = -c.y;
           c.z = -c.z;
          return c;
       }
      public static void Main(string[] args)
      {
         Console.WriteLine("\t \t Implementation of Unary operator Overloading!! \n ");
         Console.WriteLine("\t \t **********************************************");
         Negate p = new Negate (5,-6, 4);
         p.display();
         Negate p1 = new Negate (5, -6, 4);
         p1 = -p;
         Console.WriteLine("\nNegation is applied to all the values!!\n");
         p1.display();
         Console.ReadKey();
      }
   }
```

```
}
```

OUTPUT:


file:///C:/Users/BIST/Documents/Visual Studio 2012/Projects/ConsoleApplication1/ConsoleApplica...
Implementation of Unary operator Overloading!!
****
X=5,y=-6,z=4
Negation is applied to all the values!!
X=-5,y=6,z=-4

4.b.BINARY OPERATOR OVERLOADING

ALGORITHM

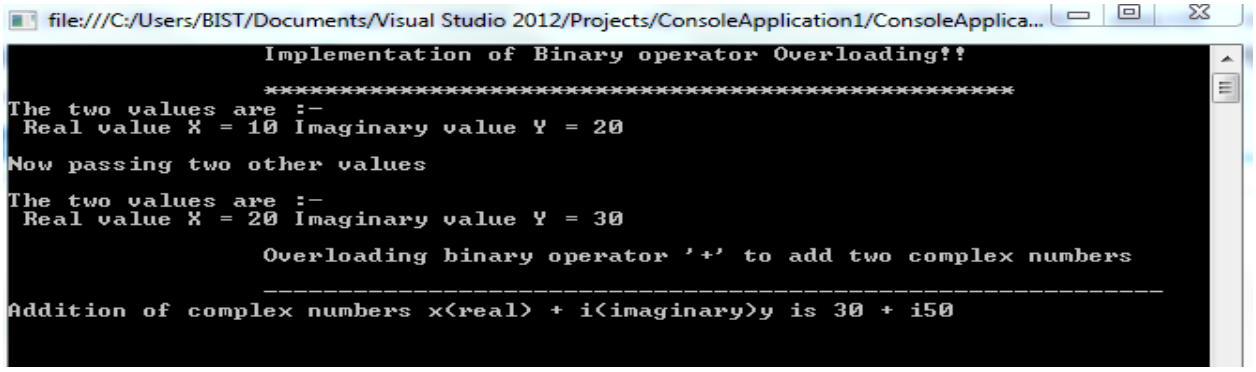
Step 1: Create a new project and a console application in Visual Studio 2012
Step 2: Create a class named “complex” and declare real and imaginary values
Step 3: Write a method for overloading (“+”) binary operator and add the real & imaginary values
Step 4: In main method, objects are created for invoking the complex class. The + operator is overloaded to add the values in the objects of Complex class
Step5: Display the values of real and imaginary and also the sum of the complex numbers

PROGRAM:

```
namespace ConsoleApplication4
{
   class Complex
   {
      decimal x, y;
      public Complex()
      {
      }
      public Complex(decimal real, decimal imag)
      {
         x = real;
         y = imag;
      }
      public void ShowXY()
      {
       Console.WriteLine("The two values are :- \n Real value X = {0} Imaginary value Y = {1} \n", x, y);
      }
      public void show()
      {
         Console.WriteLine("Addition of complex numbers x(real) + i(imaginary)y is {0} + i{1}", x, y);
      }
      public static Complex operator +(Complex c1, Complex c2)
      {
         Complex temp = new Complex();
         temp.x = c1.x + c2.x;
         temp.y = c1.y + c2.y;
         return temp;
      }
   }
   class MyClient
   {
      public static void Main()
      {
         Complex c1 = new Complex(10, 20);
         Console.WriteLine("\t \t Implementation of Binary operator Overloading!! \n ");
         Console.WriteLine("\t \t ************************************************");
         c1.ShowXY(); // displays 10 & 20
         Complex c2 = new Complex(20, 30);
         Console.WriteLine("Now passing two other values \n");
         c2.ShowXY(); // displays 20 & 30
         Complex c3 = new Complex();
         c3 = c1 + c2;
         Console.WriteLine("\t \t Overloading binary operator '+' to add two complex numbers \n");
          Console.WriteLine("\t \t ------------------------------------------------------------");
```

```
        c3.show();
        Console.ReadKey();
      }
   }
   }
```

OUTPUT:



Result:

Thus the program in C# for operator overloading concepts is implemented, executed successfully and the output is verified

EXPERIMENT -5
THREADING

AIM

To develop a C# console application to implement threading concepts

ALGORITHM:

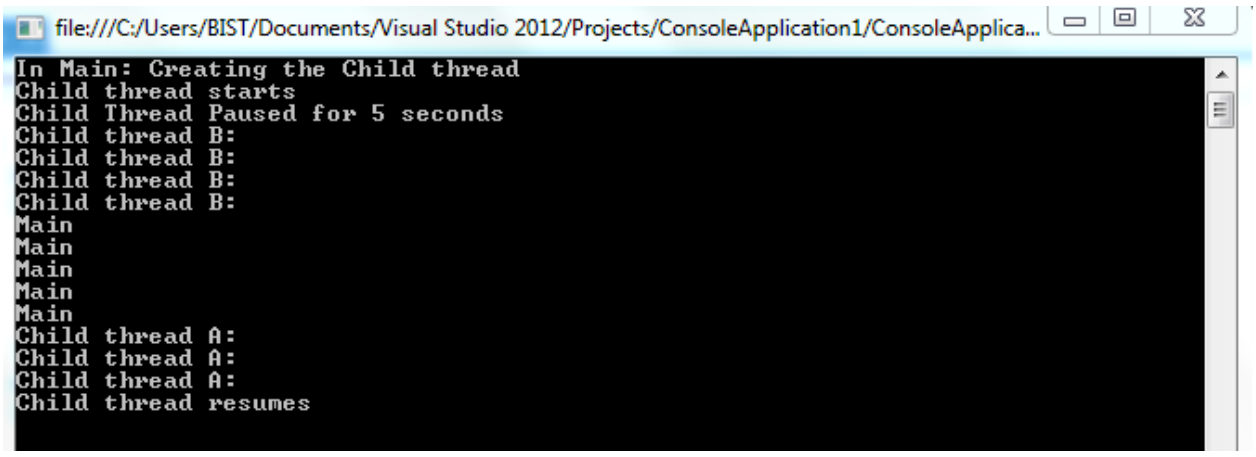
Step1 : Create a new project for developing a c# console application using visual studio 2012
Step 2: Create a class called "ThreadClass" inside "namespace -ForegroundThread"
Step3: Create methods to make a thread to sleep, pause and resume using Start, sleep, resume methods
Step 4: create two threads A & B and give highest priority to a thread to make it execute first using Thread.Priority method
Step 5: Display the thread names which are executing

PROGRAM

```
using System;
using System.Threading;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApplication1
{
   class ThreadCreationProgram
   {
      public static void ChildThread()
      {
         Console.WriteLine("Child thread starts");

         // the thread is paused for 5000 milliseconds
         int sleepfor = 5000;

         Console.WriteLine("Child Thread Paused for {0} seconds", sleepfor / 1000);
         Thread.Sleep(sleepfor);
         Console.WriteLine("Child thread resumes");
      }

      static void Main(string[] args)
      {
         ThreadStart childref = new ThreadStart(ChildThread);
         Console.WriteLine("In Main: Creating the Child thread");

         Thread t = new Thread(childref);
         t.Start();
         Console.ReadKey();

      Thread threadA= new Thread(new ThreadStart(ChildThreadA));
          // Create ThreadB object
      Thread threadB= new Thread(new ThreadStart(ChildThreadB));

         threadA.Name= "Thread A";
         threadB.Name= "Thread B";
```

```
        // set highest priority of threadB
        threadB.Priority= ThreadPriority.Highest;
        threadA.Start();
        threadB.Start();
        Thread.CurrentThread.Name= "Main";
        for (int i=0; i<5; i++)
        {
            Console.WriteLine(Thread.CurrentThread.Name);
        }
        Console.ReadKey();
    }
    public static void ChildThreadA()
    {
        for (int i=0; i<3; i++)
        {
            Console.WriteLine("Child thread A:");
        }
    }
    public static void ChildThreadB()
    {
        for (int i=0; i<4; i++)
        {
            Console.WriteLine("Child thread B:");
        }
        Console.ReadKey();
    }
}
}
```

OUTPUT



Result:

Thus the program in C# for threading concept is implemented, executed successfully and the output is verified

EXPERIMENT 6 – DELEGATES & EVENTS

AIM

To develop a c# console application to implement the following concepts:

(a) Delegates

(b) Events

(a) **DELEGATES**

ALGORITHM

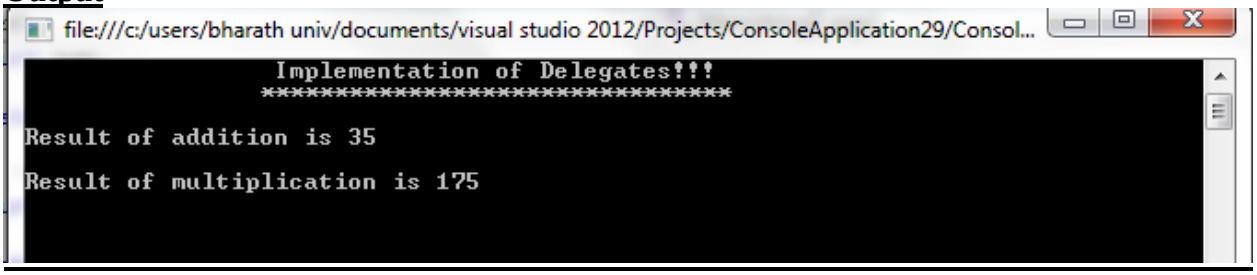
Step 1: Create a new Visual C# project →Console application
Step 2: Create a class named as 'Ex5' and declare static variable 'num' to retain its value for addition and multiplication operations
Step 3: Create a delegate named as "NumberChanger" and declare globally
Step 4: Define methods to add and multiply numbers and name it as 'Addnum" and 'multnum'
Step 5: In main method, create two objects for delegates as nc1, nc2
Step 6: addnum and multnum methods are passed as arguments within nc1 and nc2
Step 7: print the addition and multiplication result

PROGRAM:

```
namespace ConsoleApplication1
{
   delegate int NumberChange(int n);
   class ex5
   {
      static int num = 10;
      public static int addnum(int p)
      {
         num += p;
         return num;
      }
      public static int multnum(int q)
      {
         num *= q;
         return num;
      }
      public static int getnum()
      {
         return num;
      }
      static void Main(string[] args)
      {
         Console.WriteLine("\t \t Implementation of Delegates!!!");
         Console.WriteLine("\t \t**********************************\n");
         NumberChange nc1 = new NumberChange(addnum);
         NumberChange nc2 = new NumberChange(multnum);
         nc1(25);
         Console.WriteLine("Result of addition is {0}\n", getnum());
         nc2(5);
         Console.WriteLine("Result of multiplication is {0}", getnum());
         Console.ReadKey();
      }
```

```
    }
}
```

Output


file:///c:/users/bharath univ/documents/visual studio 2012/Projects/ConsoleApplication29/Consol...
Implementation of Delegates!!!
****
Result of addition is 35
Result of multiplication is 175

(b) **EVENTS**

ALGORITHM

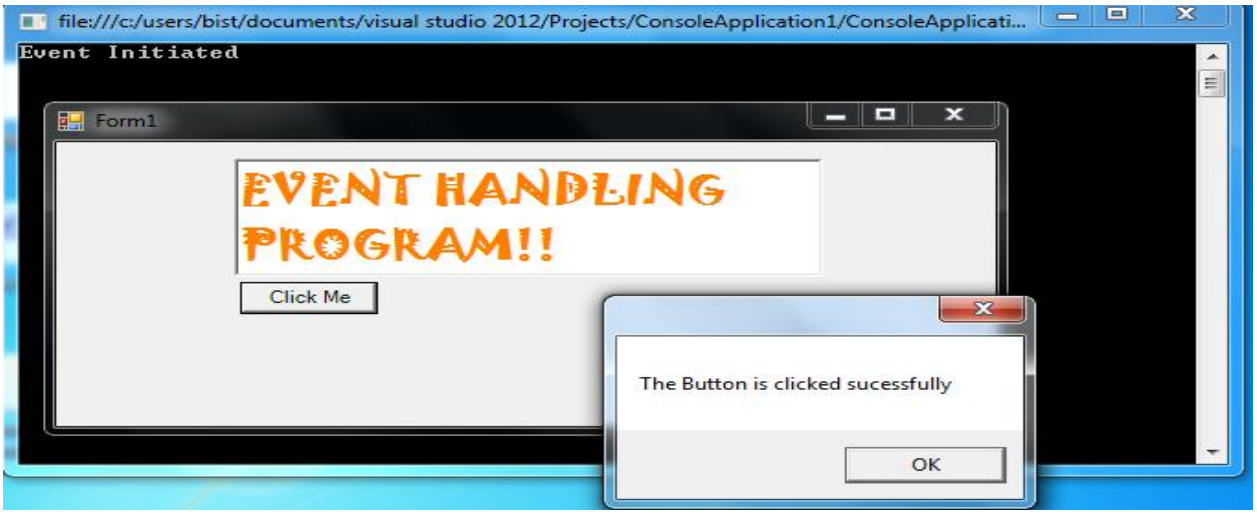
Step 1: Create a new Visual C# project →Console application
Step 2: Create a class named as 'PROGRAM' and declare delegate named as 'DelEventHandler()'
Step 3: Create a new form for the same project by selecting menu → project →Add Windows form →Add and inherit that form within this code by class Program : Form1
Step 4: insert a textbox and add contents like Event handling program
Step 5: In coding, create a button using button object
Step 5: create an event handling method so that when the button is clicked, it displays a message box
Step 6: Console application displays the message as 'Event initiated' when the button event is created successfully.
Step 7: Save the program and execute!

PROGRAM:

```
Using System;
using System.Drawing;
using System.Windows.Forms;
namespace ConsoleApplication2
{
   public delegate void DelEventHandler();

   class Program : Form1
   {
      //custom event
      public event DelEventHandler add;

      public Program()
      {
         // desing a button over form
         Button btn = new Button();
         btn.Parent = this;
         btn.Text = "Click Me";
         btn.Location = new Point(100, 100);

         //Event handler is assigned to
         // the button click event
         btn.Click += new EventHandler(onClcik);
         add += new DelEventHandler(Initiate);

         //invoke the event
         add();
      }
      //call when event is fired
      public void Initiate()
      {
         Console.WriteLine("Event Initiated");
      }
      //call when button clicked
      public void onClcik(object sender, EventArgs e)
      {
```

```
        MessageBox.Show("The Button is clicked sucessfully");
    }
    static void Main(string[] args)
    {
        Application.Run(new Program());
        Console.ReadLine();
    }
  }
}
```

OUTPUT:



Result:

Thus the program in C# for delegates and events concepts is implemented, executed successfully and the output is verified

EXPERIMENT-7
WINDOWS FORM CONTROL

AIM

To design a window based application using C# code in VB.Net

ALGORITHM

1. Create a new project and select Visual C# → Windows Form application
2. Select controls like labels, buttons, textboxes, picturebox, progressbar etc from toolbox and design a form based on your application
3. Double click on controls and add necessary C# coding
4. Save and Run the application

PROGRAM

FORM1.CS

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace swcourses
{
   public partial class Form1 : Form
   {
      Form4 f4 = new Form4();
      Form3 f3 = new Form3();
      Form2 f2 = new Form2();
     private void linkLabel2_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)
      {
         f3.Show();
         this.WindowState = FormWindowState.Minimized;
      }

    private void linkLabel3_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)
      {
         f4.Show();
         this.WindowState = FormWindowState.Minimized;
      }

    private void linkLabel1_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)
      {
         f2.Show();
         this.WindowState = FormWindowState.Minimized;

      }

       }
}
```

FORM2.CS

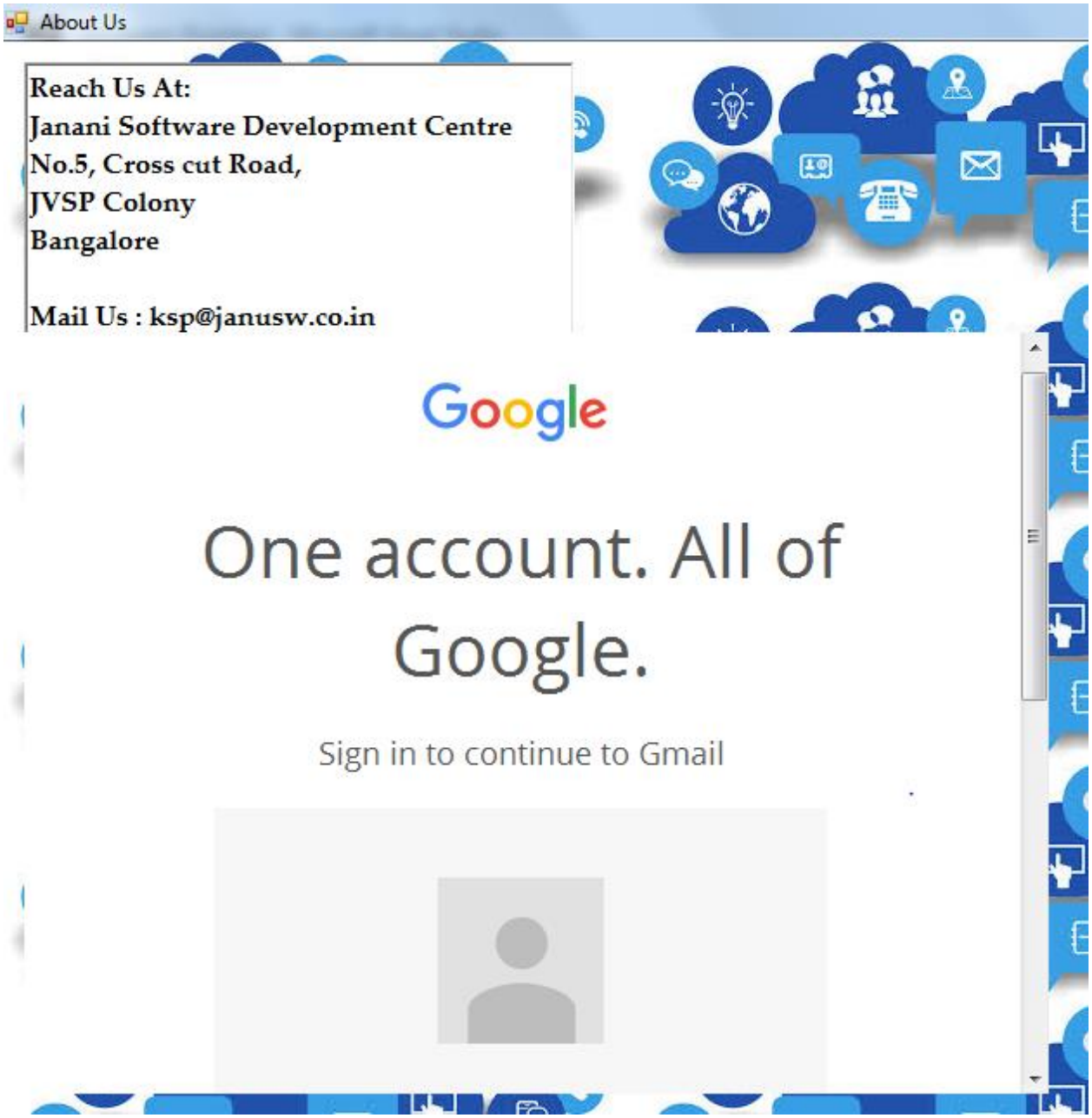
```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace swcourses
{

   public partial class Form2 : Form
   {
      public Form2()
      {
         InitializeComponent();
      }


      private void button1_Click(object sender, EventArgs e)
      {
     MessageBox.Show("Registration Successful!!, Welcome to our Centre!! All the
                  Best!");

      }
   }
}
```

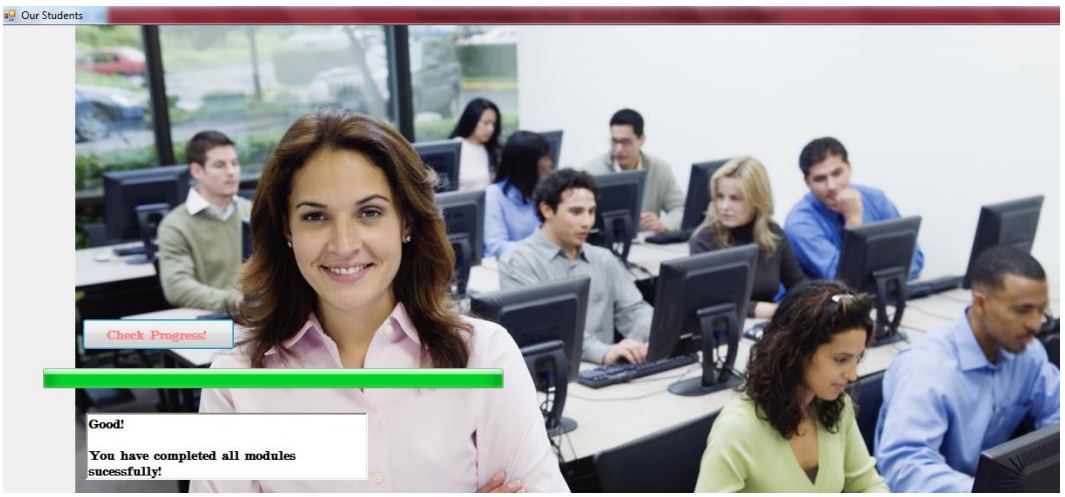
FORM3.CS

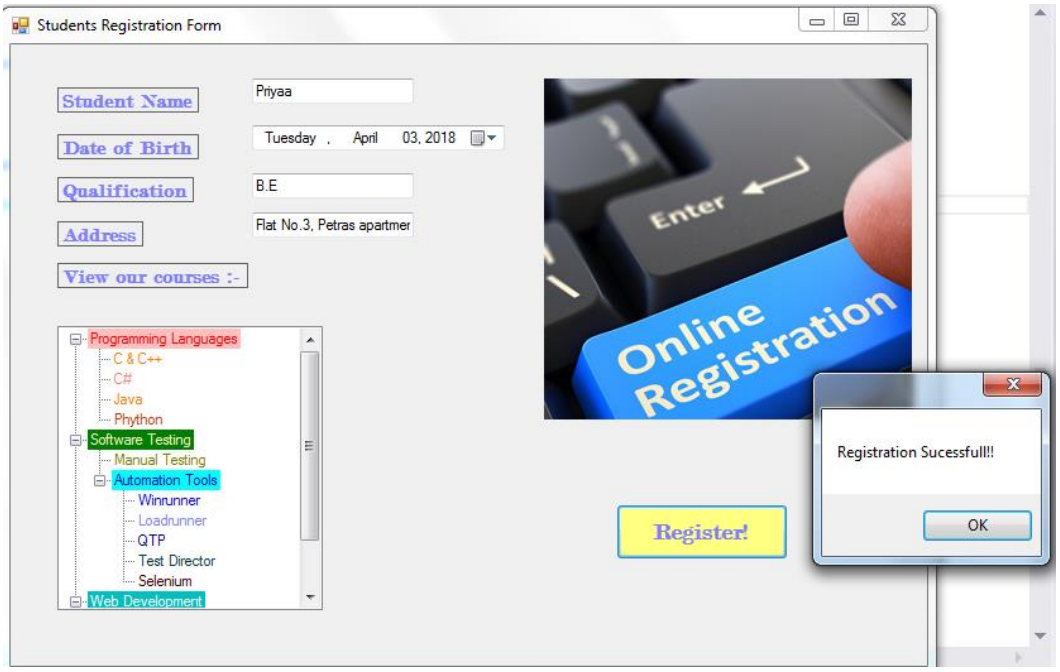
```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
namespace swcourses
{
   public partial class Form3 : Form
   {
      ProgressBar pBar = new ProgressBar();

        public Form3()
      {
         InitializeComponent();
      }

           private void button1_Click(object sender, EventArgs e)
      {
         progressBar1.Visible = true;
```

```
        pBar.Dock = DockStyle.Bottom;
        int i;

        progressBar1.Minimum = 0;
        progressBar1.Maximum = 200;

        for (i = 0; i <= 200; i++)
        {
           progressBar1.Value = i;
        }
        richTextBox1.Visible = true;
      }
      private void Form3_Load(object sender, EventArgs e)
      {
        richTextBox1.Visible = false;

        progressBar1.Visible = false;

      }
   }
}
```

OUTPUT


About Us
Reach Us At:
Janani Software Development Centre
No.5, Cross cut Road,
JVSP Colony
Bangalore
Mail Us : ksp@janusw.co.in
Google
One account. All of Google.
Sign in to continue to Gmail





Result:

Thus the program in C# for windows form control concept is implemented, executed successfully and the output is verified

EXPERIMENT -8
VALIDATING DATA

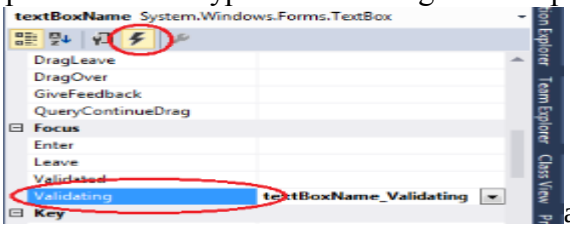
AIM

To implement validating data entered in controls using

(a) Windows based application – Manual coding for validation

(b) Web based application – Validation Controls

ALGORITHM

1. Create a new project → Windows Form application →View →Toolbox
2. Place 2 Labels, 2 Textboxes and 2 buttons in form
3. Add error provider control from toolbox in the form to display error during validation
4. In textbox's properties, select Error on error provider 1 and type the error msg to be displayed
5. Select textbox1 and right-click →properties 

and double click the marked property in the picture and type the coding
6. Type the coding to validate the data entered in the textboxes.
7. Save and run the project
8. Create a new project →ASP.Net Web Application
9. Click Project →Add New Item →Web Form →Add
10. Click Design tab at the bottom of the page and add 6 labels, 5 textboxes, 1 calendar control
11. From toolbox, select validation to see the list of validation controls available
12. Select Required Field validator, compare validator, Range validator and Regular expression validator controls and place it near textboxes and change properties to display error message as needed
13. Add the following code in .aspx page to allow only alphabets and . for passenger name textbox → `ValidationExpression="[a-zA-Z ]*."`
14. Add validation summary control at the end to display all errors in form
15. Save and run in internet explorer or Google chrome

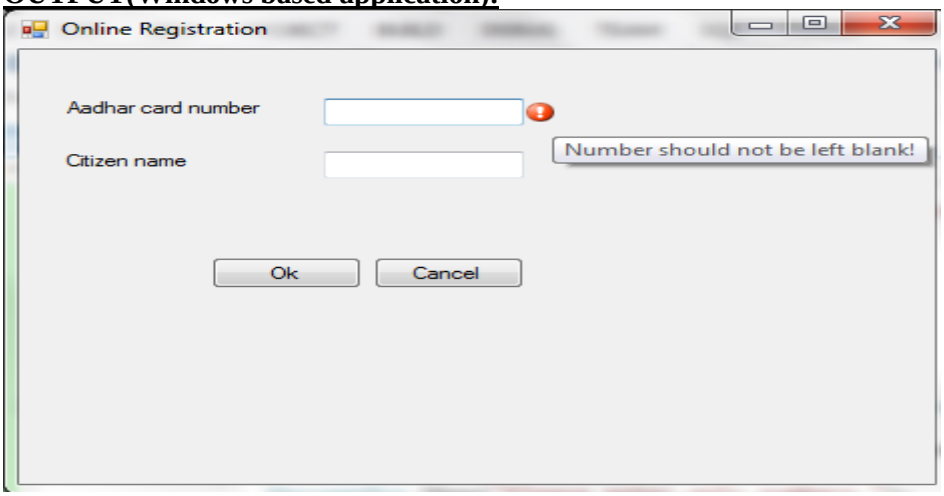
PROGRAM

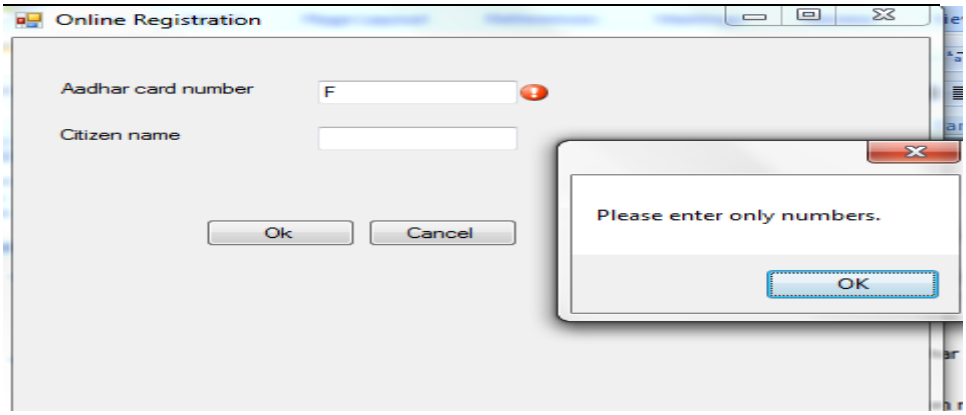
CODE(Windows based application):

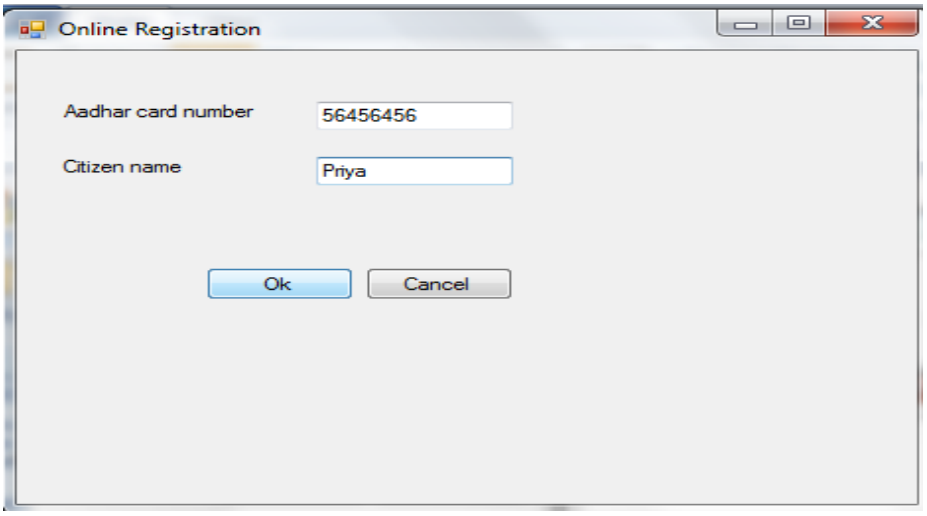
```
private void textBox1_Validating(object sender, CancelEventArgs e)
{
   if (string.IsNullOrWhiteSpace(textBox1.Text))
   {
      e.Cancel = true;
      textBox1.Focus();
      errorProvider1.SetError(textBox1, "Number should not be left blank!");
   }
   else
   {
      e.Cancel = false;
      errorProvider1.SetError(textBox1, "");
   }
}
```

```
private void textBox1_KeyPress(object sender, KeyPressEventArgs e)
{
   if (System.Text.RegularExpressions.Regex.IsMatch(textBox1.Text, "[^0-9]"))
   {
      MessageBox.Show("Please enter only numbers.");
      textBox1.Text = textBox1.Text.Remove(textBox1.Text.Length - 1);
   }
}
```

OUTPUT(Windows based application):







Result:

Thus the program in C# for validating data is implemented, executed successfully and the output is verified

EXPERIMENT – 9

CUSTOM DIALOG BOX & MDI APPLICATION

AIM

To design a notepad application to implement menus, custom dialog box and MDI concepts

ALGORITHM

1. Create a new project → Visual C# → Windows application
2. From Toolbox, include Richtextbox, Menustrip , Color dialog, Font dialog, OpenFileDialog and savefiledialog controls
3. After including menustrip controls, type the required menus like File,new,open,save, Edit,cut,copy,paste, clear, format, font,color,help
4. Form1 ->Rich text box and menus , Form2 →Custom Dialogbox and Form3 →Include richtextbox and webbrowser control
5. Double-click menu items and add the code
6. Click Project→Add item →Windows form and insert rich text box and web browser controls and type webaddress in URL property of webbrowser control
7. Click Form1→properties-.IsMDIContainer property →set to true
8. In Form1→Window state property →Maximized
9. Select richtextbox in form1 and form3 and select its property 'Dock' and select top/bottom
10. In Form2→include label and 2 buttons and Select the form and change its properties →FormBorderStyle→select as Fixed Dialog to create a custom dialog box

PROGRAM

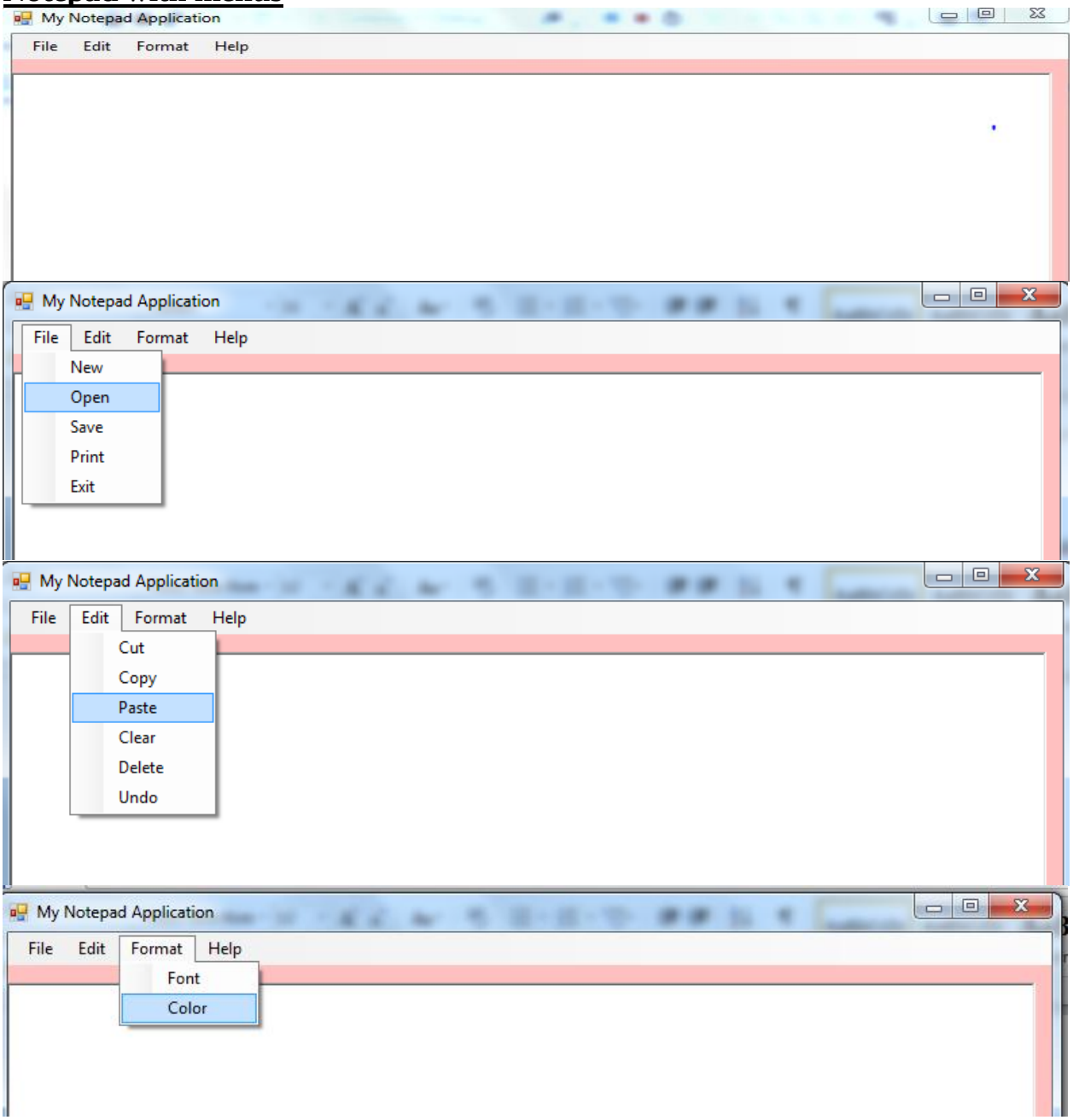
```
using System;
using System.Windows.Forms;

namespace WindowsFormsApplication11
{
   public partial class Form1 : Form
   {
      public Form1()
      {
         InitializeComponent();
      }

      private void openToolStripMenuItem_Click(object sender, EventArgs e)
      {
         OpenFileDialog dlg = new OpenFileDialog();
         dlg.ShowDialog();

         if (dlg.ShowDialog() == DialogResult.OK)
         {
```

```
        string fileName;
        fileName = dlg.FileName;
        MessageBox.Show(fileName);
    }
}

private void colorToolStripMenuItem_Click(object sender, EventArgs e)
{
    ColorDialog dlg = new ColorDialog();
    dlg.ShowDialog();

     richTextBox1.ForeColor = dlg.Color;
}

private void fontToolStripMenuItem_Click(object sender, EventArgs e)
{

    FontDialog fdlg = new FontDialog();
    fdlg.ShowDialog();
    richTextBox1.Font = fdlg.Font;
  }

private void exitToolStripMenuItem_Click(object sender, EventArgs e)
{
    PrintDialog dlg = new PrintDialog();
    dlg.ShowDialog();
}

private void saveToolStripMenuItem_Click(object sender, EventArgs e)
{
    SaveFileDialog sfdlg = new SaveFileDialog();
    sfdlg.Filter = "Text Files (*.txt) | *.txt";
    string Saved_File = " ";
    saveFileDialog1.InitialDirectory = "C:";
    saveFileDialog1.FileName = "";
    if (sfdlg.ShowDialog() == DialogResult.OK)
    {

        Saved_File = saveFileDialog1.FileName;
        richTextBox1.SaveFile(Saved_File, RichTextBoxStreamType.RichText);
    }

}
Form2 f2 = new Form2();
private void deleteToolStripMenuItem_Click(object sender, EventArgs e)
{
    f2.Show();
    DialogResult res = MessageBox.Show("Are you sure you want to Delete",
   "Confirmation", MessageBoxButtons.OKCancel, MessageBoxIcon.Information);
    if (res == DialogResult.OK)
    {
        MessageBox.Show("Are you sure? ");
```

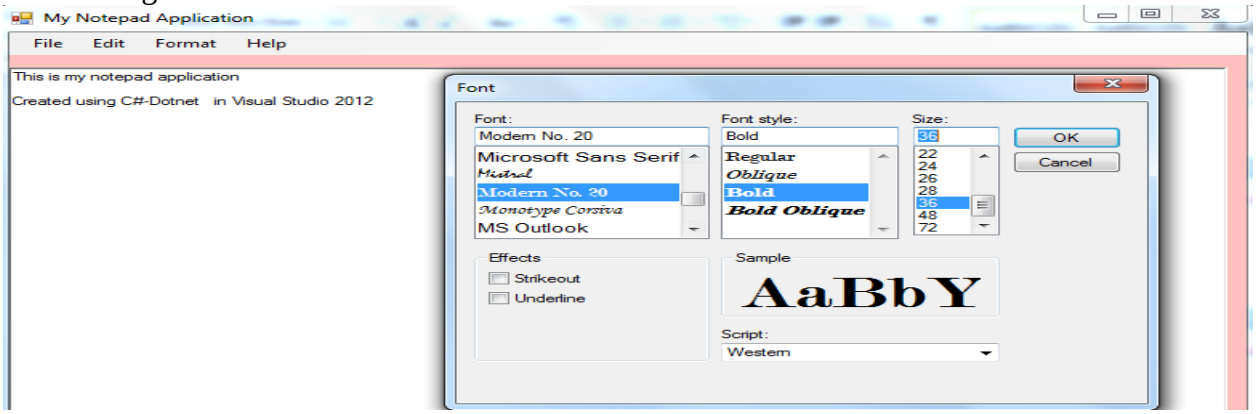
```
            //Some task…
        }
        if (res == DialogResult.Cancel)
        {
            this.Close();
            //Some task…
        }
    }

    private void copyToolStripMenuItem_Click(object sender, EventArgs e)
    {
        richTextBox1.Copy();

    }

    private void clearToolStripMenuItem_Click(object sender, EventArgs e)
    {
        richTextBox1.Text = " ";
        Clipboard.Clear();
    }

    private void pasteToolStripMenuItem_Click(object sender, EventArgs e)
    {
        richTextBox1.Paste();
    }

        }
}
```

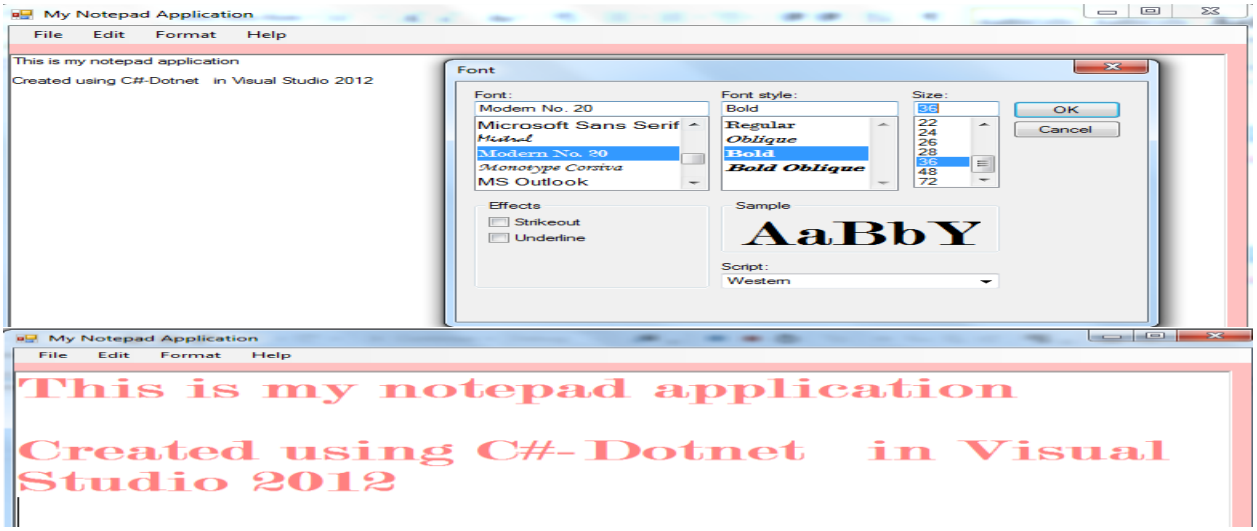
OUTPUT

Notepad with menus

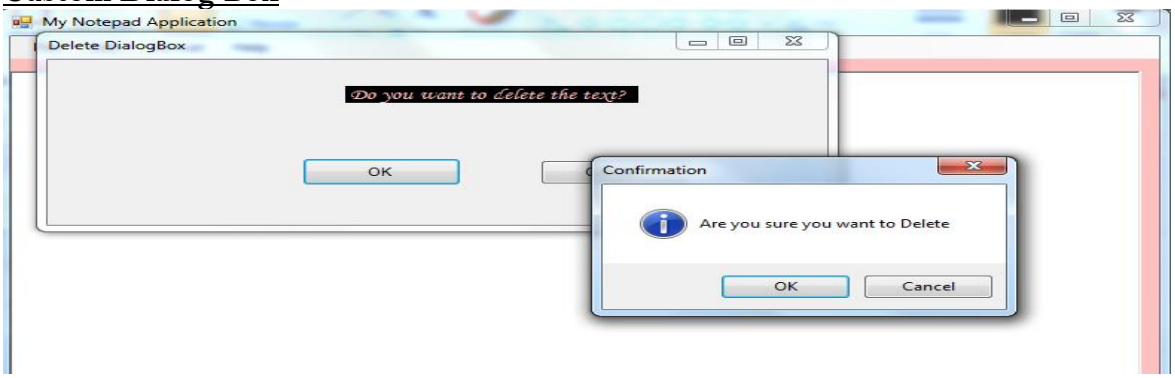


Font Dialog Box

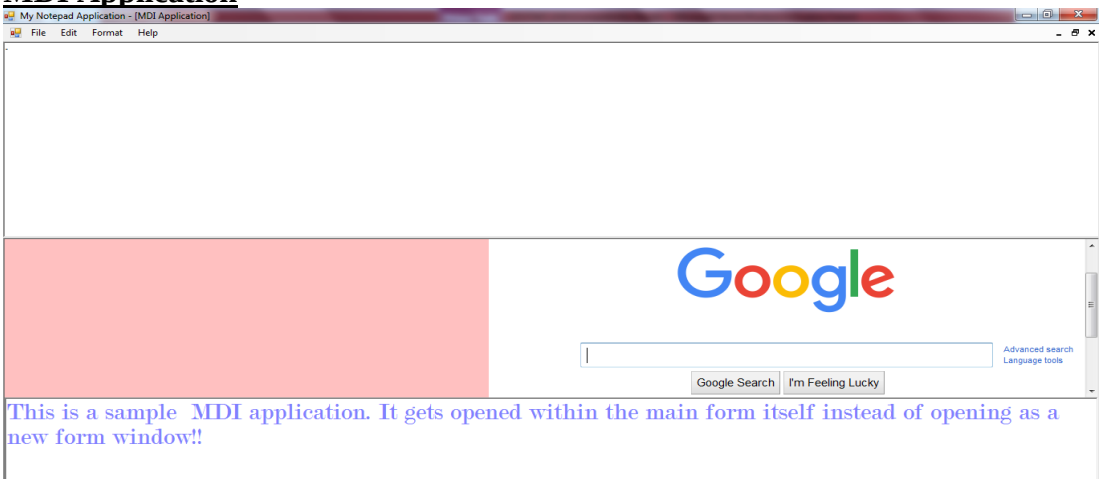




Custom Dialog Box



MDI Application



Result:

Thus the program in C# for custom dialog box and MDI is implemented, executed successfully and the output is verified

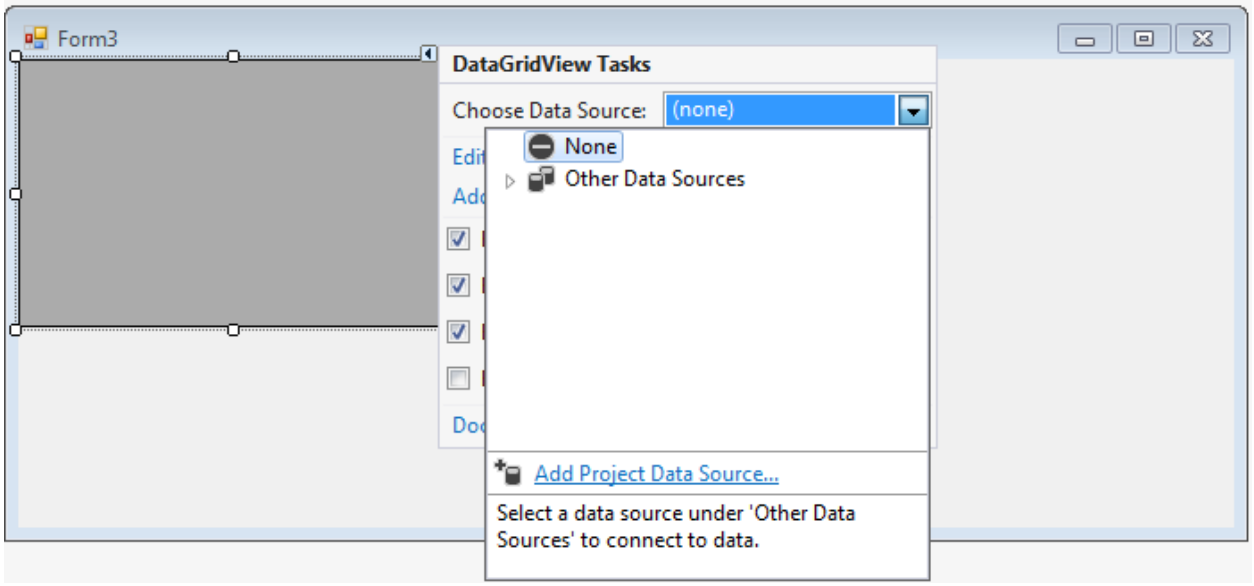
EXPERIMENT – 10
RETRIEVING DATA FROM DATABASE & WORKING WITH DISCONNECTED ENVIRONMENT

AIM

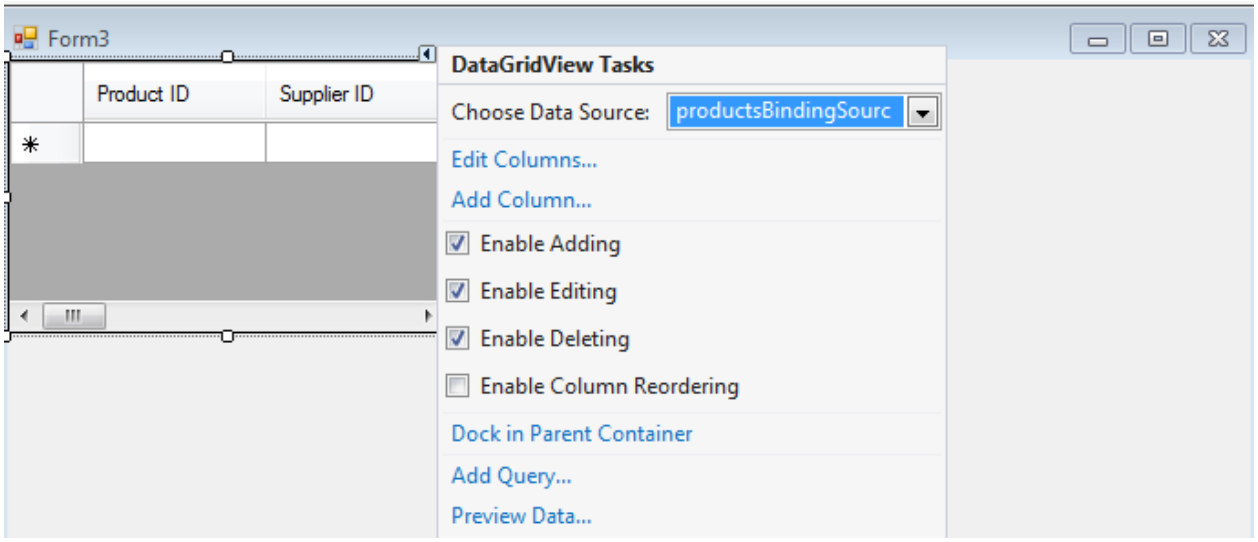
To design windows based application to retrieve data from SQL database and to work with disconnected environment in ADO.Net using C#

ALGORITHM

- Create a new project → Windows Application →Name →ok
- Design your form with necessary labels and pictures
- From toolbox, select "DatagridView" control and place it in form
- Select Add project Data Source from this:



- Select database and select dataset, click next, click new connection and click change button and select Microsoft SQL Server Compact 4.0 → ok button
- Click browse button and select Northwind and select open button
- Click Test connection button and click ok
- Select Next → Yes button
- Double-click Tables folder to view the list of tables available for the northwind database
- Check products checkbox and uncheck all other checkbox to display products table in the windows form
- Click Finish button and select Add query button from this page:



- Click Query Builder button → Execute Query → Ok →ok button
- Run the application

PROGRAM

Form1 Coding

```
private void button1_Click(object sender, EventArgs e)
{
   Form2 f2 = new Form2();
   f2.Show();
}
```

Form2 Coding:

```
private void Form2_Load(object sender, EventArgs e)
{
        this.productsTableAdapter.Fill(this.ex10.Products);

}
```

OUTPUT

Result:

Thus the program in SQL for retrieving database application is implemented, executed successfully and the output is verified