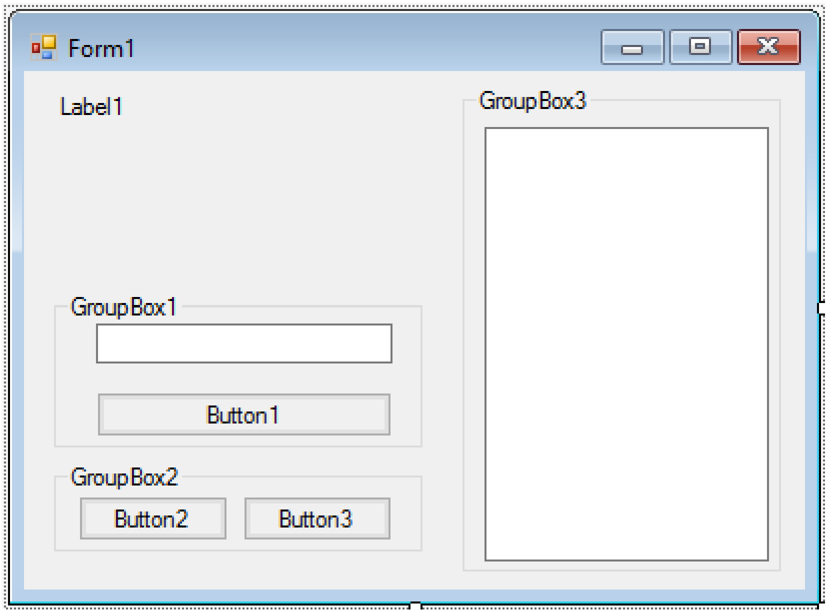
Project 7 - Decode

Project Design

This project is a classic computer game. The computer generates a four-digit code (with no repeating digits). You guess at the code. The computer then tells you how many digits in your guess are correct and how many digits are in the correct location. Based on these clues, you make a new guess. You continue guessing until you have cracked the code. The project you are about to build is saved as **Decode** in the project folder (**\BeginVCS\BVCS Projects**).

Place Controls on Form

Start a new project in Visual C#. Place a label control (set **AutoSize** to **False** so it can be resized), two button controls and two group box controls on the form. In the first group box, place a text box control and a button. In the second group box, place a text box control. When done, your form should look something like this:



Set Control Properties

Set the control properties using the properties window:

Form1 Form:

Property Name	**Property Value**
Text	Decode
FormBorderStyle	FixedSingle
StartPosition	CenterScreen

label1 Label:

Property Name	**Property Value**
Name	lblMessage
Text	[Blank]
TextAlign	MiddleCenter
BorderStyle	Fixed3D
BackColor	White
ForeColor	Blue
Font Size	12

groupBox1 Group Box:

Property Name	**Property Value**
Name	grpGuess
BackColor	Red
Text	[Blank]
Visible	False

textBox1 Text Box:

Property Name	**Property Value**
Name	txtGuess
TextAlign	Center
Font	Courier New
Font Size	16
MaxLength	4

button1 Button:

Property Name	**Property Value**
Name	btnCheck
Text	Check Guess
Font	Arial
Font Size	12
BackColor	Light Yellow

button2 Button:

Property Name	**Property Value**
Name	btnNew
Text	New Game

button3 Button:

Property Name	**Property Value**
Name	btnStop
Text	Stop
Enabled	False

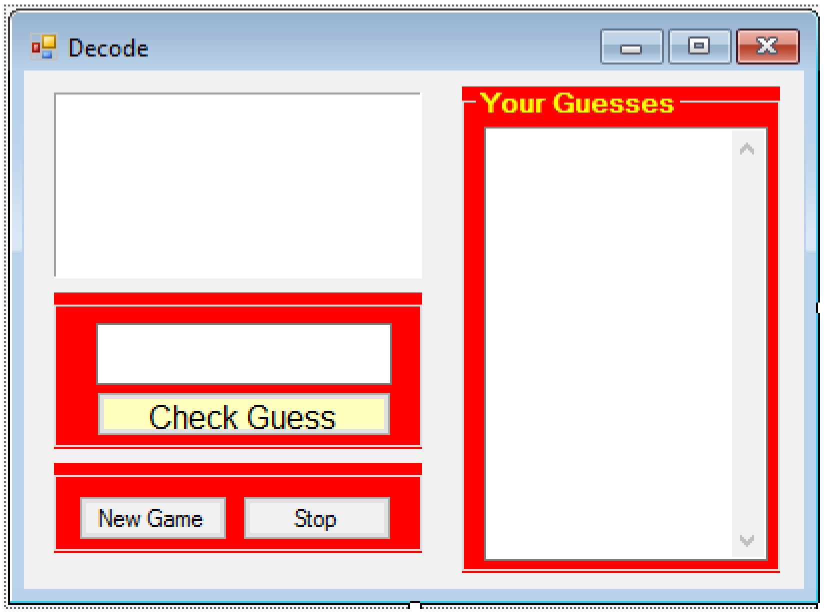
groupBox2 Group Box:

Property Name	**Property Value**
Name	grpGuesses
Text	Your Guesses
BackColor	Red
ForeColor	Yellow
Font Size	10
Font Style	Bold

textBox2 Text Box:

Property Name	**Property Value**
Name	txtGuesses
Font	Courier New
Font Size	14
BackColor	White
MultiLine	True
ReadOnly	True
TabStop	False
ScrollBars	Vertical

When done setting properties, my form looks like this:



Write Event Methods

Most of the code in this project is involved with generating a four-digit computer code (when you click **New Game**) and checking the guess you input (click **Check Guess**). The **Your Guesses** group box provides a history of each guess you made.

Add this code to the **general declarations** area:

```
bool gameOver;
string computerCode;
string[] computerNumbers = new string[4];
Random myRandom = new Random();
```

The **Form1_Load** event method:

```
private void Form1_Load(object sender, EventArgs e)
{
    lblMessage.Text = "Click New Game";
    btnNew.Focus();
}
```

The **btnNew_Click** event method:

```
private void btnNew_Click(object sender, EventArgs e)
{
    string[] nArray = new string[10];
    int i, j;
    string t;
    // Start new game
    btnStop.Enabled = true;
    gameOver = false;
    lblMessage.Text = "";
    txtGuesses.Text = "";
    txtGuess.Text = "";
    btnNew.Enabled = false;
    // Choose code using modified version of card shuffling
routine
    // Order all digits initially
    computerCode = "";
    for (i = 0; i < 10; i++)
    {
        nArray[i] = Convert.ToString(i);
    }
    // J is number of integers remaining
    for (j = 9; j >=6; j--)
    {
        i = myRandom.Next(j);
        computerNumbers[9 - j] = nArray[i];
        computerCode = computerCode + nArray[i];
        t = nArray[j];
        nArray[j] = nArray[i];
        nArray[i] = t;
    }
    lblMessage.Text = "I have a 4 digit code.\r\nTry to guess
it.";
    grpGuess.Visible = true;
    txtGuess.Focus();
}
```

The **btnStop_Click** event method:

```
private void btnStop_Click(object sender, EventArgs e)
{
    // Stop current game
    grpGuess.Visible = false;
    btnNew.Enabled = true;
    btnStop.Enabled = false;
    if (!gameOver)
    {
        lblMessage.Text = "Game Stopped\r\nMy code was - " +
computerCode;
    }
    btnNew.Focus();
}
```

The **txtGuess_KeyPress** event method:

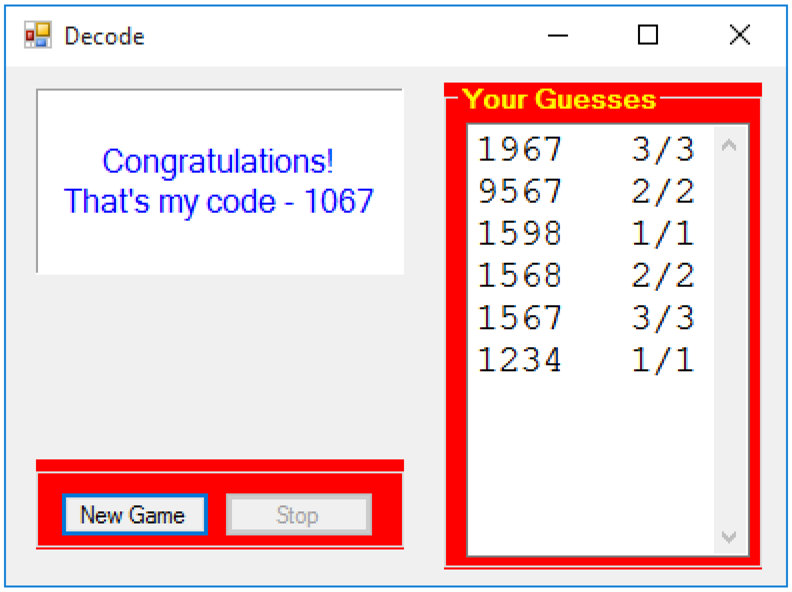
```
private void txtGuess_KeyPress(object sender,
KeyPressEventArgs e)
{
    // Allow numbers only
    if ((int) e.KeyChar == 13)
    {
        btnCheck.PerformClick();
    }
    else if ((e.KeyChar >= '0' && e.KeyChar <= '9') || (int)
e.KeyChar == 8)
    {
        e.Handled = false;
    }
    else
    {
        e.Handled = true;
    }
}
```

The **btnCheck_Click** event method:

```
private void btnCheck_Click(object sender, EventArgs e)
{
    string w;
    string[] wNumbers = new string[4];
    int i, j, k1, k2;
    bool distinct;
    // Check your guess
    w = txtGuess.Text;
    // Check length validity
    if (w.Length != 4)
    {
        lblMessage.Text = "Guess must have 4 numbers
...\r\nTry again!";
        txtGuess.Focus();
        return;
    }
    else
    {
        // Get numbers and make sure they are distinct
        distinct = true;
        for (i = 0; i < 4; i++)
        {
            wNumbers[i] = w.Substring(i, 1);
            if (i != 0)
            {
                for (j = i - 1; j >= 0 ; j--)
                {
                    if (wNumbers[i] == wNumbers[j])
                    {
                        distinct = false;
                    }
                }
            }
        }
        if (!distinct)
        {
            lblMessage.Text = "Numbers must all be different
...\r\nTry again!";
            txtGuess.Focus();
            return;
        }
        if (w == computerCode)
        {
            lblMessage.Text = "Congratulations!\r\nThat's my
code - " + computerCode;
            gameOver = true;
            btnStop.PerformClick();
            return;
        }
        else
        {
            // Compute score
            k1 = 0;
            k2 = 0;
            for (i = 0; i < 4; i++)
            {
                for (j = 0; j < 4; j++)
                {
                    if (wNumbers[j] == computerNumbers[i])
                        k1++;
                }
                if (wNumbers[i] == computerNumbers[i])
                    k2++;
            }
            lblMessage.Text = "Your guess - " + w + "\r\n" +
Convert.ToString(k1) + " digit(s) correct\r\n" +
Convert.ToString(k2) + " digit(s) in proper place";
            txtGuesses.Text = w + "   " +
Convert.ToString(k1) + "/" + Convert.ToString(k2) + "\r\n" +
txtGuesses.Text;
            txtGuess.Text = "";
            txtGuess.Focus();
        }
    }
}
```

Run the Project

Save your work. Run the project. Click **New Game** to start. Type a guess for the four-digit code. Note the computer will not let you type an illegal guess (non-distinct, less than 4 digits). Click **Check Guess**. After each guess, the computer will tell you how many digits are correct and how many are in the correct location. For your reference, a history of your guesses is displayed under **Your Guesses**. The score is displayed as two numbers separated by a slash. The first number is the number of correct digits, the second the number in the correct location. Click **Stop** to stop guessing and see the computer's code. Here's a game I played:



Congratulations!
That's my code - 1067

```
1967    3/3
9567    2/2
1598    1/1
1568    2/2
1567    3/3
1234    1/1
```

Other Things to Try

You can give this game a variable difficulty by allowing the user to choose how many digits are in the code, how many numbers are used to generate the code, and whether digits can repeat. See if you can code up and implement some of these options. The commercial version of this game (called MasterMind) uses colored pegs to set the code. This makes for a prettier game. See if you can code this variation.

Lastly, many mathematical papers have been written on developing a computer program that can decode the kinds of codes used here. Do you think you could write a computer program to determine a four digit code you make up? The program would work like this: (1) computer makes a guess, (2) you tell computer how many digits are correct and how many are in correct locations, then, (3) computer generates a new guess. The computer would continue guessing until it gave up or guessed your code.