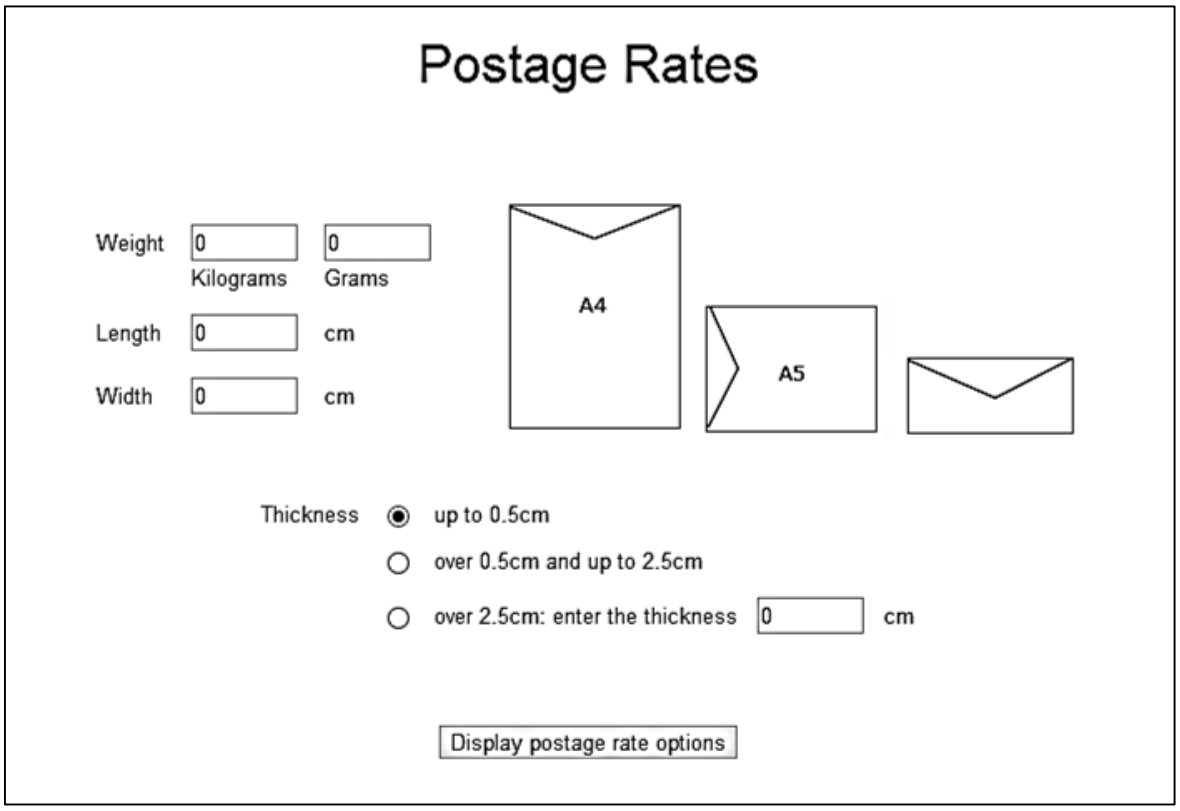
2 Postage Rates

In this project we will create a website page which will allow the user to check the postage rate for a letter or parcel. The postage rates at the time of writing are shown in the table on the next page.

Postage depends on the size of the item being sent, its weight, and whether first class or second class delivery is required. In order to calculate the cost, we will design an input page to carry out the following functions:

- Allow the weight of the item to be entered as kilograms and grams.
- Allow the length and width of the item to be entered in centimetres. For common letter sizes, it will be easiest for the user to click on a picture image of the envelope. The program will then insert the corresponding length and width values into the input boxes automatically.
- Allow the thickness of the item to be entered. This will be given as a choice of three thickness bands. For thickness over 2.5cm, the actual thickness measurement will also be required.

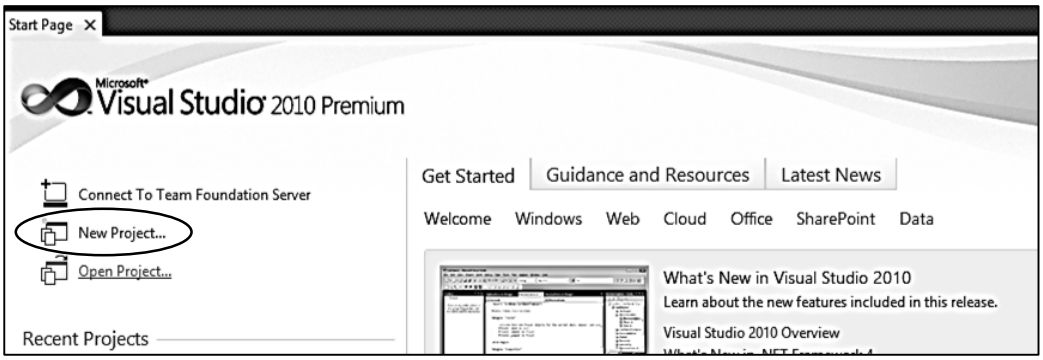


When the 'Display postage rate options' button is clicked, the costs for first and second class post (where applicable) will be displayed. The user can then make a choice as to which service they wish to use.

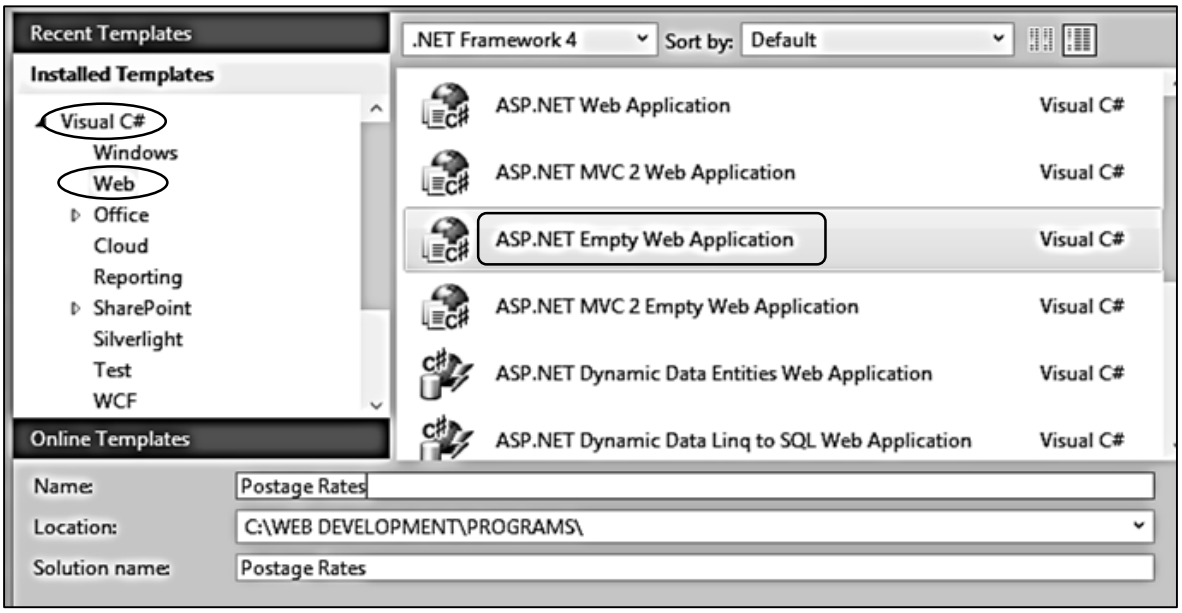
Letters and postcards				
	Royal Mail 1st Class		**Royal Mail 2nd Class**	
	Letters	Large Letters	Letters	Large Letters
0-100g	60p	90p	50p	69p
101-250g	-	£1.20	-	£1.10
251-500g	-	£1.60	-	£1.40
501-750g	-	£2.30	-	£1.90
Letters	Size (up to) Length: 24cm, Width: 16.5cm, Thickness: 0.5cm, Weight (up to) 100g			
Large Letters	Size (up to) Length: 35.3cm, Width: 25cm, Thickness: 2.5cm, Weight (up to) 750g			

Parcels					
	Royal Mail 1st Class		**Royal Mail 2nd Class**		
	Small Parcels	Medium Parcels	Small Parcels	Medium Parcels	Large Parcels
up to 1kg	£3.00	£5.65	£2.60	£5.20	-
up to 2kg	£6.85	£8.90	£5.60	£8.00	£11.96
up to 5kg	-	£15.10	-	£13.35	£12.92
up to 10kg	-	£21.25	-	£19.65	£15.92
up to 15kg	-	£32.40	-	£27.70	£22.46
up to 20kg	-	£32.40	-	£27.70	£27.68
up to 25kg	-	-	-	-	£38.48
up to 30kg	-	-	-	-	£42.50
Small Parcels	Size (up to) Length: 45cm, Width: 35cm, Depth: 8cm, Weight (up to) 2kg				
Medium Parcels	Size (up to) Length: 61cm, Width: 46cm, Depth: 46cm, Weight (up to) 20kg				
Large Parcels	For Parcels greater in size than 61cm x 46cm x 46cm or heavier than 20kg.				

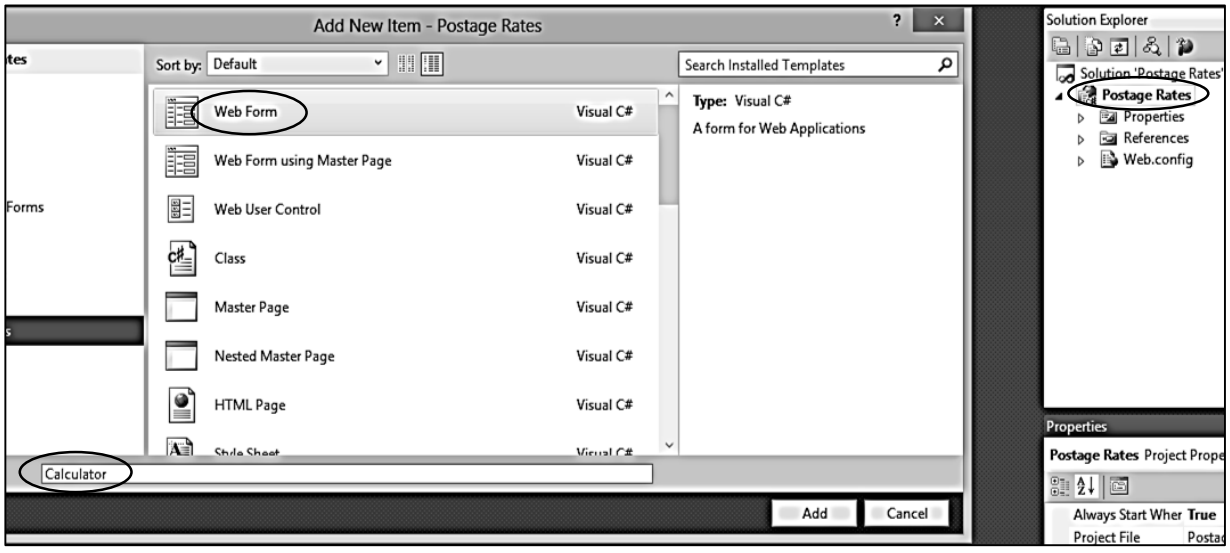
Start a new project:



Select ***Visual C# Web***, and click on the ***ASP.NET Empty Web Application*** option. Choose a location to store your project, and give the name '***Postage Rates***'.



The empty web site project will be created. Right click the ***Postage Rates*** project icon, then select ***Add / New item***. Click on ***Web Form***, and give the name '***Calculator***'.



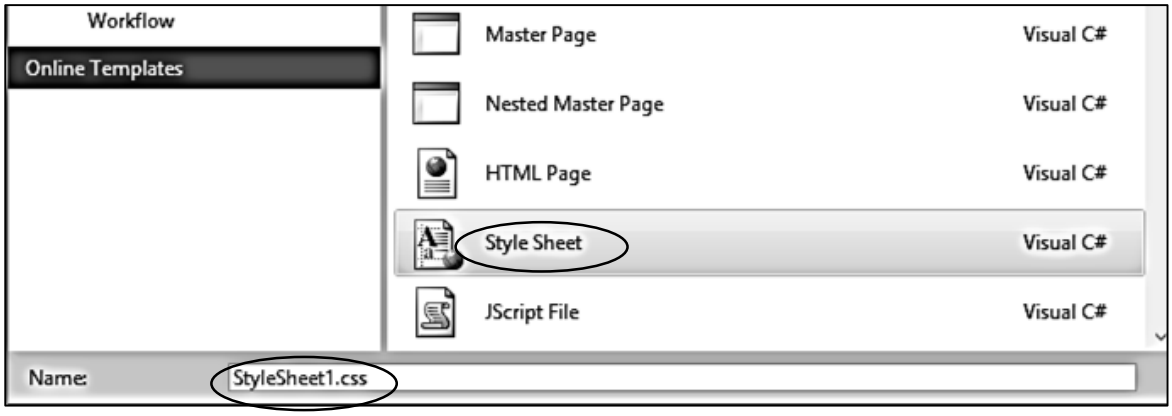
The ***Calculator*** HTML page will open. Add a title '***Postage Calculator***' for the page tab when the website runs. Give an id name for the "***content***" division, and add a main heading '***Postage Rates***' at the top of the page:

```
<head runat="server">
    <title>Postage Calculator</title>
</head>
<body>
    <form id="form1" runat="server">
    <div id="content">
       <br />
       <h1>Postage Rates</h1>
       <br />
    </div>
    </form>
</body>
</html>
```

Click the 'Design' button to view the page so far…

We can imporove the appearance by centering the title and using a different font. To do this, we will need a style sheet.

Go to the ***Soution Explorer*** window, right click the ***Postage Rates*** project icon and select ***Add / New item***. Choose ***Style Sheet***, and accept the name '***StyleSheet1***'.



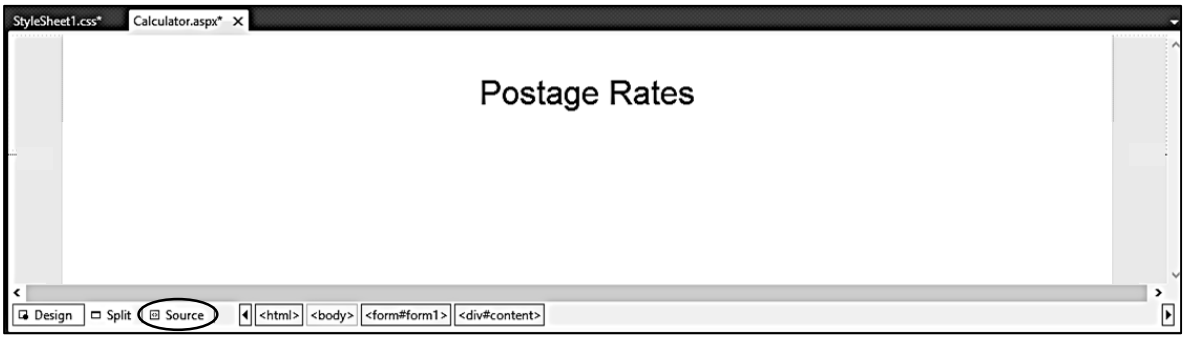
Add code to the style sheet:

```
body
{
    background: #E9E9E9;
    font-family: Arial, Helvetica, sans-serif;
    margin: 0px;
    padding: 0px;
}

#content
{
  width:1000px;
  height:800px;
  background-color: White;
  margin-left: auto;
  margin-right: auto;
  color: Black;
}

h1
{
    text-align: center;
    font-size: xx-large;
    font-weight:normal;
}
```

Return to the ***Calculator*** HTML page and add a link to the style sheet in the ***<head>*** section of the page:

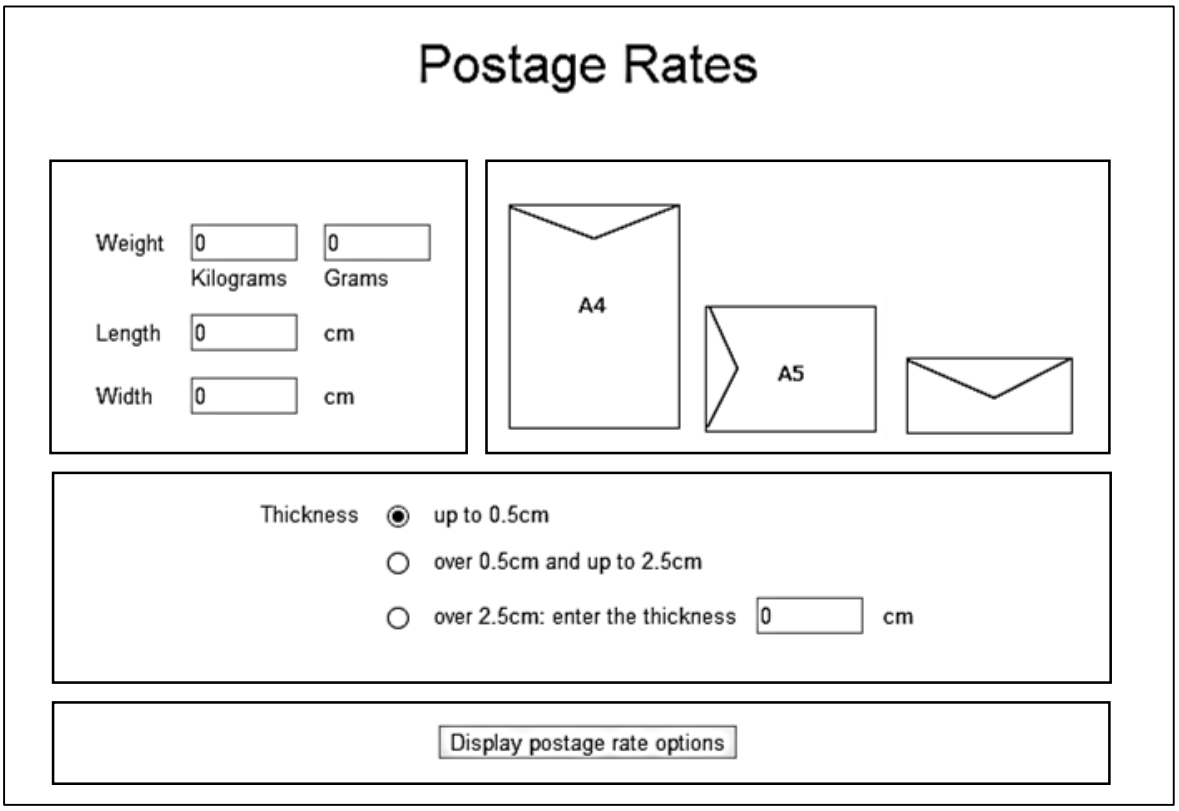
```
<head runat="server">
    <title>Postage Calculator</title>
    <link rel="Stylesheet" type="text/css" href="StyleSheet1.css" />
</head>
```

Use the ***Design*** button option to see a preview of the page. We have created a content area with a width of 1000 pixels. This has a white background, and is centred on the grey background of the screen. The page heading is now also centred, and is displayed in a sans-serif font.



Click the '***Source***' button to return to the HTML page.

The next step is to add components for input of the size and weight of the postal item. It will be convenient to lay out the screen using a series of divisions. These will provide a neat set of boxes into which we can insert components.



Add code to the '***content***' division. Each of the new subdivisions has an ID name which describes its purpose.

```
<body>
   <form id="form1" runat="server">
   <div id="content">
       <br />
       <h1>Postage Rates</h1>

       <div id="weightAndSize">
       </div>

       <div id="envelopeOptions">
       </div>

       <div id="thickness">
       </div>

       <div id="buttonAndResultsOutput">
       </div>
    </div>
    </form>
</body>
```

We require as the top two divisions, ***weightAndSize*** and ***envelopeOptions*** to be alongside one another, with the remaining two divisions taking the full width of the page below. To arrange this, go to the style sheet and add formatting code for the divisions.

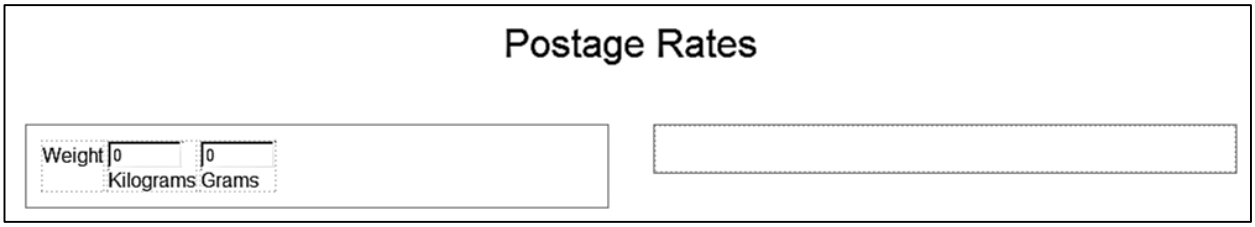
```
h1
{
    text-align: center;
    font-size: xx-large;
    font-weight:normal;
}

#weightAndSize
{
    margin: 10px;
    float: left;
    width: 450px;
    padding: 10px;
    border: 1px solid #bbb;
}

#envelopeOptions
{
    margin: 10px;
    float: right;
    width: 450px;
    padding:10px;
       border:1px solid #bbb;
}

#thickness
{
    margin: 10px;
    float:left;
    width: 960px;
    padding:10px;
       border:1px solid #bbb;
}

#buttonAndResultsOutput
{
    margin: 10px;
    float:left;
    width: 960px;
    padding:10px;
       border:1px solid #bbb;
}
```

Return to the design view. The first two divisions are now arranged alonside each other, with the remaining divisions underneath, as we require.

We can now work on the first ***division*** where weights and sizes are input. This section can be laid out as a table so that the components are neatly aligned. We will construct this table in the ***weightAndSize*** division. Insert code to input weight.

```
<div id="weightAndSize">
 <table>
      <tr>
         <td>
            Weight
            <br />
             
         </td>
         <td>
             <asp:TextBox ID="txtKilos" runat="server" Width="60" Text="0">
             </asp:TextBox>
             <br />
             Kilograms
        </td>
        <td>
             <asp:TextBox ID="txtGrams" runat="server" Width="60" Text="0">
             </asp:TextBox>
             <br />
             Grams
        </td>
      </tr>
 </table>
</div>
```

Click the ***Design*** button to go to preview window.

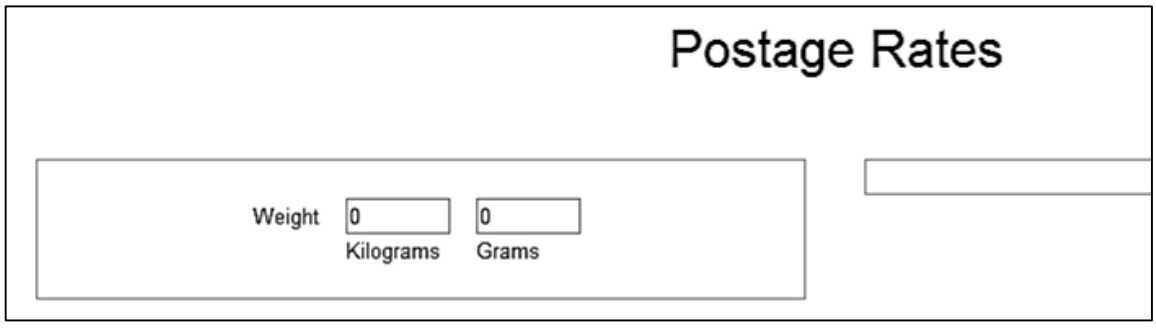


We have created the first line of the input table successfully, but it would be better if this was centred on the page and cells were separated more. We will also make the font size slightly smaller.

Go to the style sheet and add an entry for ***table***. Also add a line to the ***body*** section to adjust the font size.

```
body
{
    background: #E9E9E9;
    margin: 0px;
    padding: 0px;

    font-size: .80em;
}

table
{
    margin-left: auto;
    margin-right : auto;
    border-spacing: 10px;
}

#content
{
  width:1000px;
  background-color: White;
```

Go back to the Design screen and see the effects of these changes. The layout now looks better.



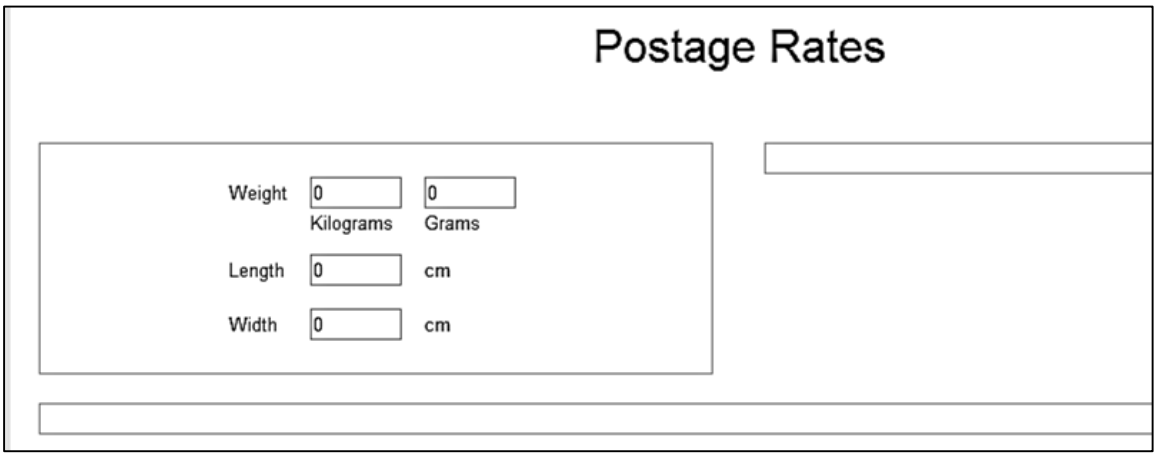
Return to the HTML page and add code to input the length and width of the postal item.

```
            Grams
        </td>
    </tr>
    <tr>
        <td>
            Length
        </td>
        <td>
            <asp:TextBox ID="txtLength" runat="server" Width="60" Text="0">
            </asp:TextBox>
        </td>
        <td>
            cm
        </td>
    </tr>
</table>
```

Check that the design view now displays the input box and labels correctly for length.
Complete this section of the form by adding similar lines of code to input the width of the postal item, as shown:

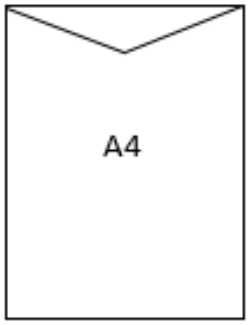
```
            <asp:TextBox ID="txtLength" runat="server" Width="60" Text="0">
            </asp:TextBox>
        </td>
        <td>
            cm
        </td>
    </tr>
    <tr>
        <td>
            Width
        </td>
        <td>
            <asp:TextBox ID="txtWidth" runat="server" Width="60" Text="0">
            </asp:TextBox>
        </td>
        <td>
            cm
        </td>
    </tr>
</table>
```

Build and run the web page to view the appearance in the web browser.

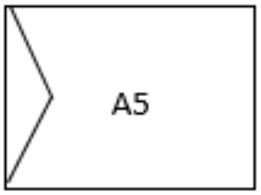


Close the web browser and return to ***Visual Studio***. Select ***Debug*** from the main menu and click the ***Stop Debugging*** option.

This completes the Weight and Length input section of the form. We will now add Envelope Size Selection options. Begin by creating three graphics images for the different envelope designs, then save these in .PNG or .JPG format.



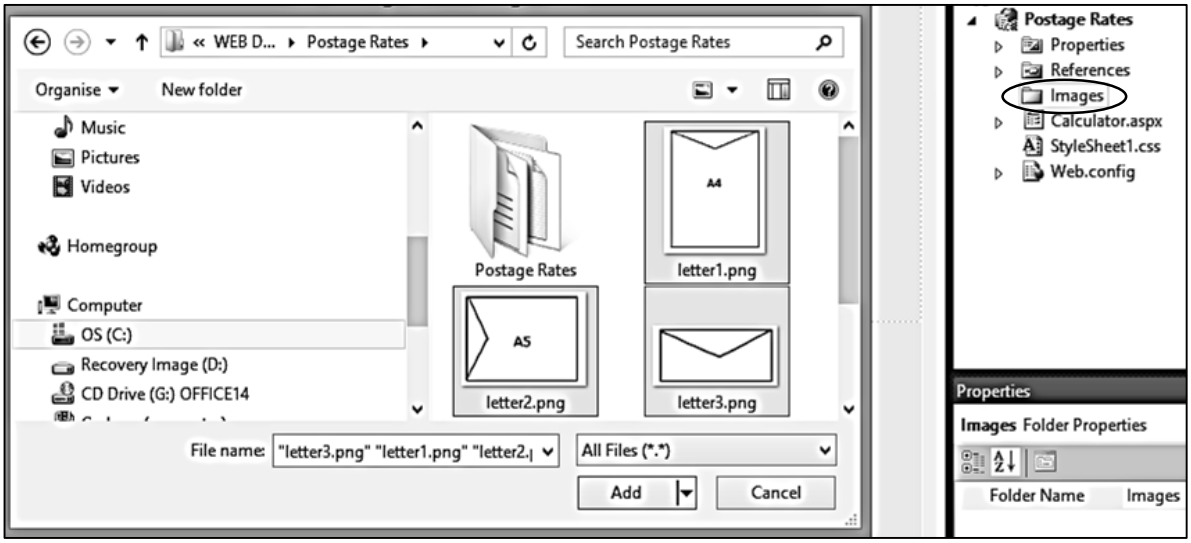
letter1.png



letter2.png

letter3.png

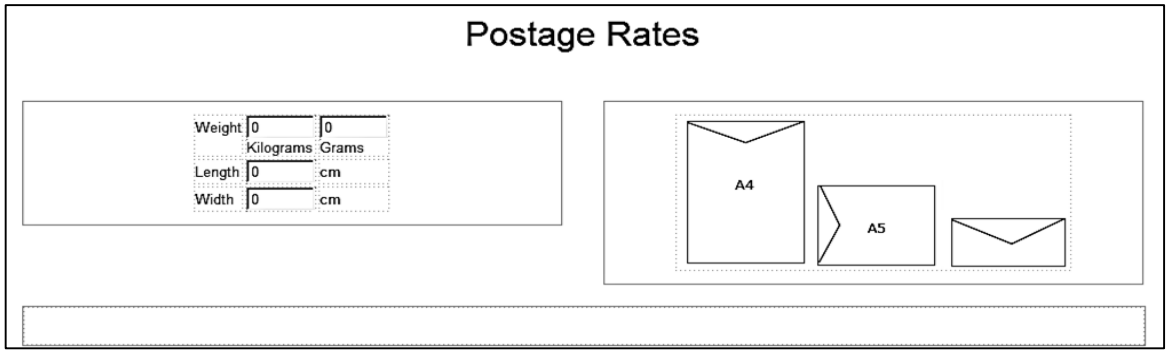
Go to the ***Solution Explorer*** window and right click the ***Postage Rates*** project icon. Select ***Add / New Folder***, and give this the name '***Images***'. Right click the Images folder and select '***Add / Existing item***'. Find the envelope images which you created, and upload these to the project.



Return to the ***Calculator*** HTML page and add code to create three image buttons to display the envelope graphics, as shown below. We will making use of a table within the ***envelopeOptions*** division.

```
<div id="envelopeOptions">
   <table>
      <tr><td>
         <asp:ImageButton ID="ImageButton1"
            ImageUrl="~/Images/letter1.png" runat="server"  />
         <asp:ImageButton ID="ImageButton2"
            ImageUrl="~/Images/letter2.png" runat="server"  />
         <asp:ImageButton ID="ImageButton3"
            ImageUrl="~/Images/letter3.png" runat="server"  />
      </td></tr>
   </table>
</div>
```

Go to the Design screen to view the layout of the components we have added so far.



We can now begin to add functionality to the application.

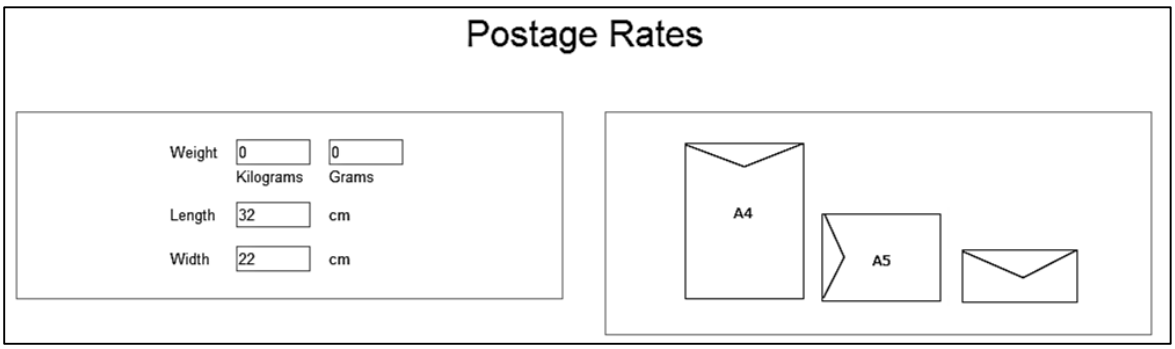
Double-click the '***A4 envelope***' image. A C# code window will open. The appearance should be familiar if you have previously worked on C# stand-alone programs. Notice that an empty ImageButton_Click method has been created, ready for you to add your own processing code.

```
namespace Postage_Rates
{
    public partial class Calculator : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
        }

        protected void ImageButton1_Click(object sender, ImageClickEventArgs e)
        {
        }
    }
}
```

The size of an A4 envelope is approximately 32cm by 22cm. Add code to the Button_Click method which will insert these measurements into the Length and Width input boxes.

```
protected void ImageButton1_Click(object sender, ImageClickEventArgs e)
{
    txtLength.Text = "32";
    txtWidth.Text = "22";
}
```

Build and run the web page. Click the A4 envelope image, and check that the measurements are inserted into the text boxes correctly. Return to ***Visual Studio*** and click the option to ***stop debugging***.



Go to the Design screen and double click the other two envelope images to create Button_Click methods, and insert code in a similar way to generate the corresponding Length and Width values.

```
protected void ImageButton1_Click(object sender, ImageClickEventArgs e)
{
    txtLength.Text = "32";
    txtWidth.Text = "22";
}

protected void ImageButton2_Click(object sender, ImageClickEventArgs e)
{
    txtLength.Text = "22";
    txtWidth.Text = "16";
}

protected void ImageButton3_Click(object sender, ImageClickEventArgs e)
{
    txtLength.Text = "22";
    txtWidth.Text = "11";
}
```

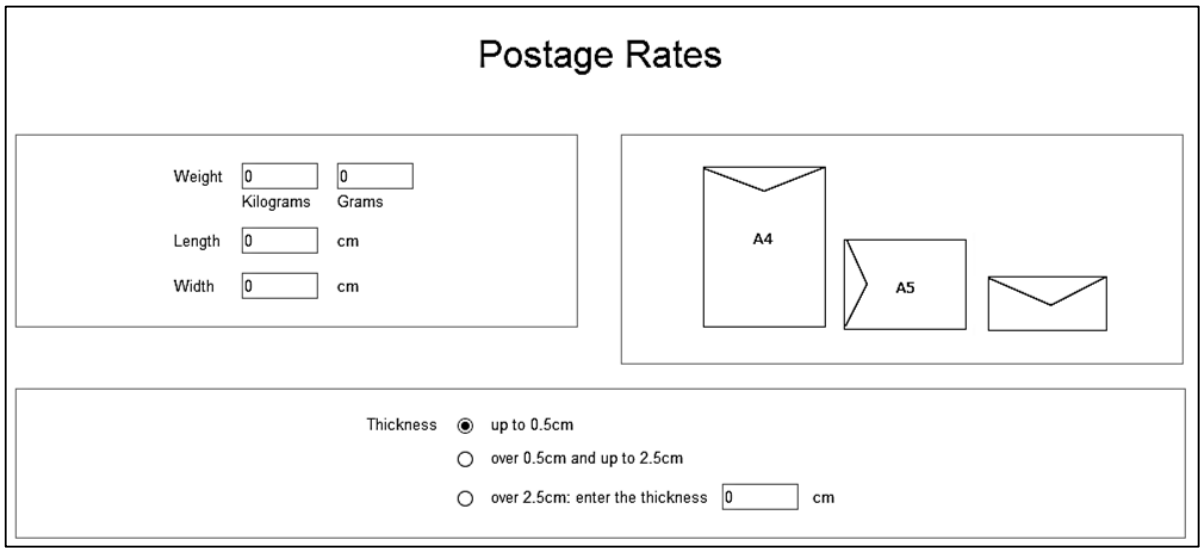
The next section to produce is a radio button group for entering the thickness of the postal item. The first two categories, up to 0.5cm and up to 2.5cm thickness, do not require exact measurements to be provided. However, for items more than 2.5cm thick it is necessary to give the actual measurement. We can again lay out the required components as a small table within a cell of the main table.

Thickness	◉ up to 0.5cm		
	○ over 0.5cm and up to 2.5cm		
	○ over 2.5cm: enter the thickness	0	cm

Go to the ***Calculator*** HTML page and add code to the ***thickness table*** division:

```
<div id="thickness">

<table>
   <tr>
      <td>
          Thickness
      </td>
      <td>
        <asp:RadioButton GroupName="thickness" ID="thin" checked="true"
               runat="server" />
      </td>
      <td>
         up to 0.5cm
      </td>
   </tr>
   <tr>
      <td></td>
      <td>
          <asp:RadioButton  GroupName="thickness" ID="medium"  runat="server" />
      </td>
      <td>
          over 0.5cm and up to 2.5cm
      </td>
   </tr>
   <tr>
      <td> </td>
      <td>
          <asp:RadioButton  GroupName="thickness" ID="thick"  runat="server" />
      </td>
      <td>
           over 2.5cm: enter the thickness
      </td>
      <td>
         <asp:TextBox ID="txtThickness" runat="server" Width="60" Text="0">
         </asp:TextBox>
      </td>
      <td>cm</td>
   </tr>
 </table>

 </div>
```

Build and run the page to see the effects of this code.

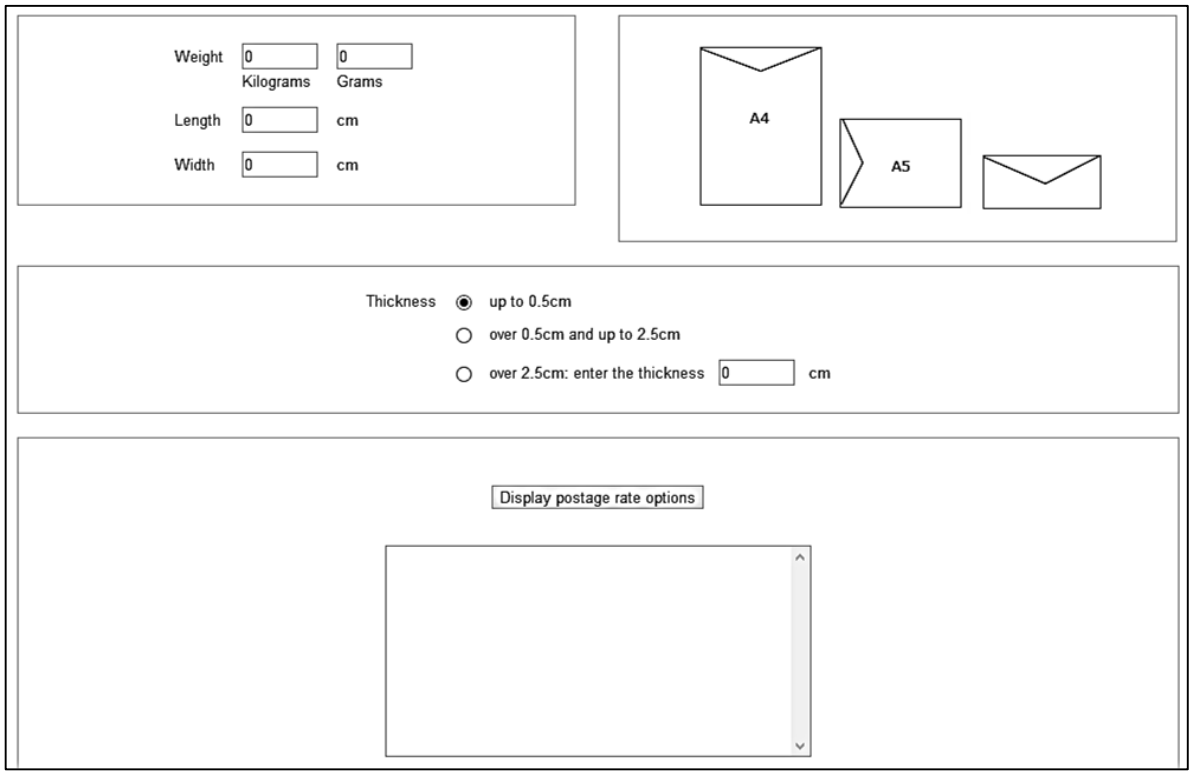


We have given all three radio buttons the same ***GroupName*** of "***thickness***" to link them together as a group, so that only one of the buttons can be selected at a time.

The final components to add to the form are: the button to carry out the calculation, and a list box for displaying the postage rates which are found. Add code to the '***button and output results'*** division of the HTML page.

```
<div id="buttonAndResultsOutput">

<table>
   <tr>
      <td align="center"><br />
          <asp:Button ID="btnCalculate"  runat="server"
                                  Text="Display postage rate options"/>
      </td>
   </tr>
   <tr>
      <td><br />
          <asp:ListBox ID="ListBox1"  runat="server" Height="180px"
                                   Width="360px">
          </asp:ListBox>
      </td>
   </tr>
</table>

</div>
```

Go to the ***Design*** preview page to see the layout of the components which have just been added.



That completes the screen design, and we can now work on the C# code to calculate the postage rates. Double click the '***Display postage rate options***' button to create a ***Button_Click*** method.

Add code to the ***Button_Click*** method which will call two other methods:

- ***checkValues( )*** is an error trapping procedure to ensure that correct data has been entered by the user.
- ***calculateCost( )*** will determine the postal rates and display these in the list box.

Add empty methods for ***checkValues( )*** and ***calculateCost( )***

```
protected void btnCalculate_Click(object sender, EventArgs e)
{
    checkValues();
    if (error == false)
    {
        calculateCost();
    }
}

protected void checkValues()
{

}

protected void calculateCost()
{

}
```

Go to the top of the C# code page and add the variables which we will need during the calculation:

```
public partial class Calculator : System.Web.UI.Page
{
    Boolean error = false;
    double weight1;
    double weight2;
    double weight;
    double length;
    double width;
    double thickness;

    protected void ImageButton1_Click(object sender, ImageClickEventArgs e)
    {
        txtLength.Text = "32";
        txtWidth.Text = "22";
    }
```

We will begin work on the error trapping procedure by setting up a try..catch block in the ***checkValues( )*** method. This will provide a general error message if text or symbols are entered instead of numbers in any of the input boxes:

```
protected void checkValues()
{
    ListBox1.Items.Clear();
    try
    {

    }
    catch
    {
        ListBox1.Items.Add("Incorrect data entered");
    }
}
```

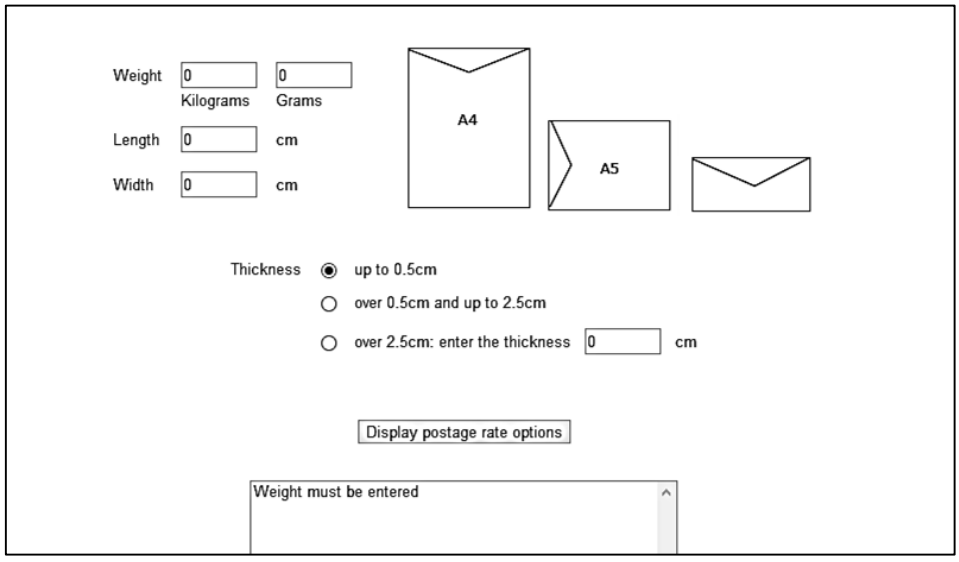
We can now check for more specific errors.

We begin by adding code to identify a missing weight value.

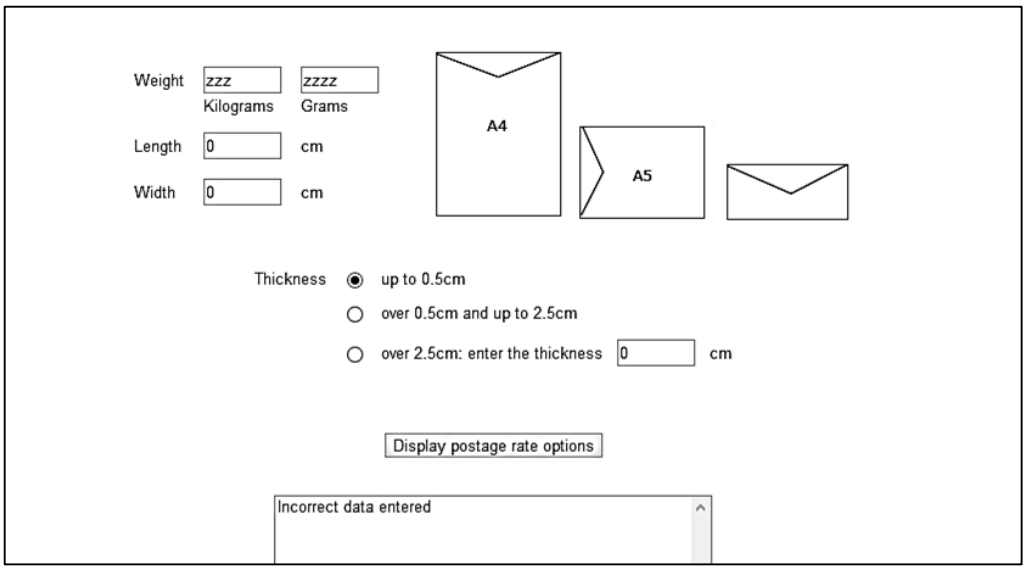
- The Kilograms or Grams input box may have been left blank by the user, for example: if the postal item was less than one kilogram in weight, or was an exact number of kilograms with no additionals grams needing to be shown. In this case, we will insert a zero figure into the empty box.
- The ***Kilogram*** and ***Gram*** textBox entries are converted to number format, then the total number of grams is calculated.
- If the weight is found to be zero, an error message is displayed in the listBox.

```
protected void checkValues()
{
    ListBox1.Items.Clear();
    try
    {
        if (txtKilos.Text == "")
        {
            txtKilos.Text = "0";
        }
        if (txtGrams.Text == "")
        {
            txtGrams.Text = "0";
        }

        weight1 = Convert.ToDouble(txtKilos.Text);
        weight2 = Convert.ToDouble(txtGrams.Text);
        weight = weight1 + (weight2 / 1000);

        if (weight == 0)
        {
            ListBox1.Items.Add("Weight must be entered");
            error = true;
        }
    }
    catch
    {
        ListBox1.Items.Add("Incorrect data entered");
    }
}
```

Build and run the web page to test the error trapping for the ***Weight*** entry. If Kilograms and Grams are both entered as zero values or left blank, an error message should appear in the list box to say that a weight must be entered.



If text characters are entered in place of numbers, then the general error message indicating incorrect data should be displayed.



Add similar code to check for blank or zero entries in the ***Length*** and ***Width*** input boxes.

```
if (weight == 0)
{
    ListBox1.Items.Add("Weight must be entered");
    error = true;
}
```

```
if (txtLength.Text == "")
{
    txtLength.Text = "0";
}

length = Convert.ToDouble(txtLength.Text);
if (length == 0)
{
    ListBox1.Items.Add("Length must be entered");
    error = true;
}

if (txtWidth.Text == "")
{
    txtWidth.Text = "0";
}

width = Convert.ToDouble(txtWidth.Text);
if (width == 0)
{
    ListBox1.Items.Add("Width must be entered");
    error = true;
}
```

We now come to the ***Thickness*** entry. The error checking is different in this case, as one of the radio buttons must have been selected. A numerical input is only required for items thicker than 2.5cm.

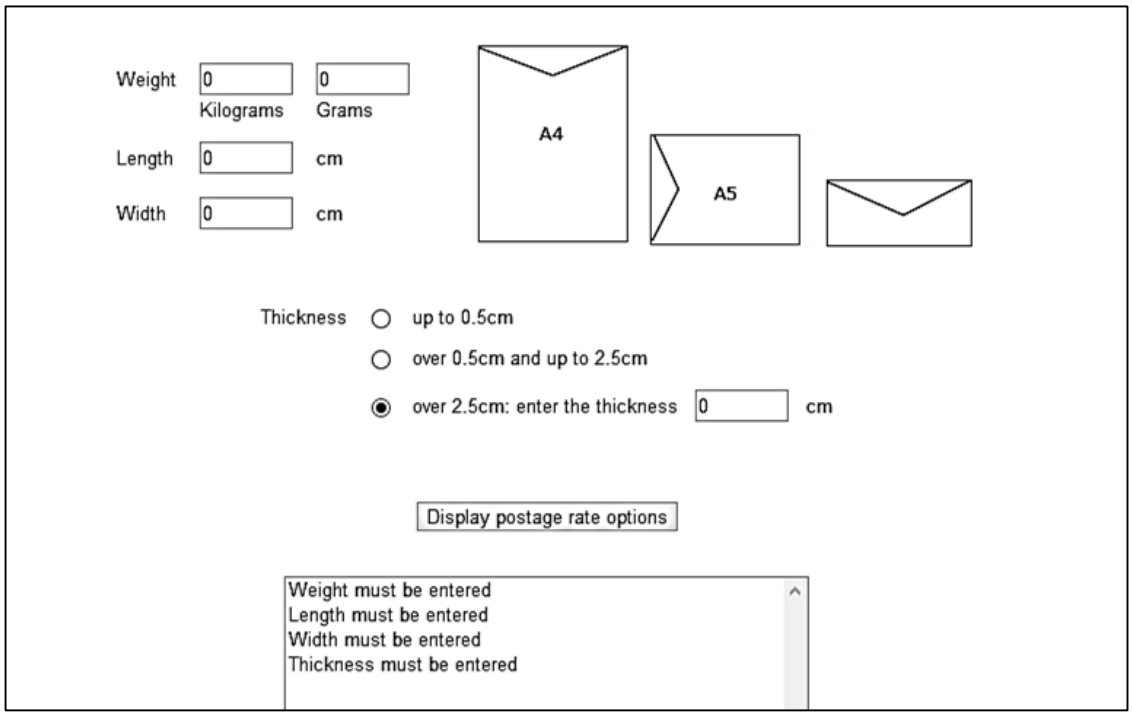
Notice that we earlier assigned the return values '**thin**', '**medium**' and '**thick**' to the three radio buttons, so the program is able to identify which button is selected.

```
<asp:RadioButton  GroupName="thickness" ID="thin"  runat="server" />
<asp:RadioButton  GroupName="thickness" ID="medium"  runat="server" />
<asp:RadioButton  GroupName="thickness" ID="thick"  runat="server" />
```

At the same time that the entries are checked, it is a good opportunity to set the numerical thickness value which will be used in the calculation of the correct postage rate. For thin items, we will assign a default value of 0.5cm, and for medium items a value of 2.5cm.

```
if (width == 0)
{
    ListBox1.Items.Add("Width must be entered");
    error = true;
}

if (thick.Checked == true)
{
    if (txtThickness.Text == "" || txtThickness.Text == "0")
    {
        ListBox1.Items.Add("Thickness must be entered");
        error = true;
    }
    else
    {
        thickness = Convert.ToDouble(txtThickness.Text);
    }
}

if (thin.Checked)
{
    thickness = 0.5;
}

if (medium.Checked)
{
    thickness = 2.5;
}
```

Compile and run the program. Check that all error messages are now displayed correctly.



If the program completes the ***checkValues( )*** method with the ***error*** variable still set to false, we can be confident that correct entries have been made for weight, length, width and thickness. The program can then proceed to the calculation of the postage rate.

Return to the C# code page and find the empty ***calculateCost( )*** method which you set up earlier. Add some preliminary lines of code. The first ***IF*** condition will exchange the ***Length*** and ***Width*** values if necessary, to ensure that ***Length*** is the largest value. We are also setting up a series of Boolean (true/false) values which will be used to identify the postage category to which the item belongs.

```
protected void calculateCost()
{
    if (width > length)
    {
        double temp = length;
        length = width;
        width = temp;
    }

    ListBox1.Items.Add("POSTAGE OPTIONS");

    Boolean letter = false;
    Boolean largeLetter = false;
    Boolean smallParcel = false;
    Boolean mediumParcel = false;
    Boolean largeParcel = false;
}
```

We can now add sections of code to check the size and weight requirements for letters. From the information in the postage rates table, a Letter must be no larger than 24cm in length, 16.5cm in width, 0.5cm in thickness, and weigh no more than 0.1Kg (100g).

We will not send the item as a ***Large letter*** if it can be sent in the cheaper ***Letter*** category.

```
Boolean mediumParcel = false;
Boolean largeParcel = false;

if (length <= 24 && width <= 16.5 && thickness == 0.5 && weight <= 0.1)
{
    letter = true;
}

if (length <= 35.3 && width <= 25 && thickness <= 2.5 && weight <= 0.75)
{
    if (letter == false)
    {
        largeLetter = true;
    }
}
```

Add similar code to identify a ***Small parcel***, ***Medium parcel*** or ***Large parcel*** by means of its weight and size.

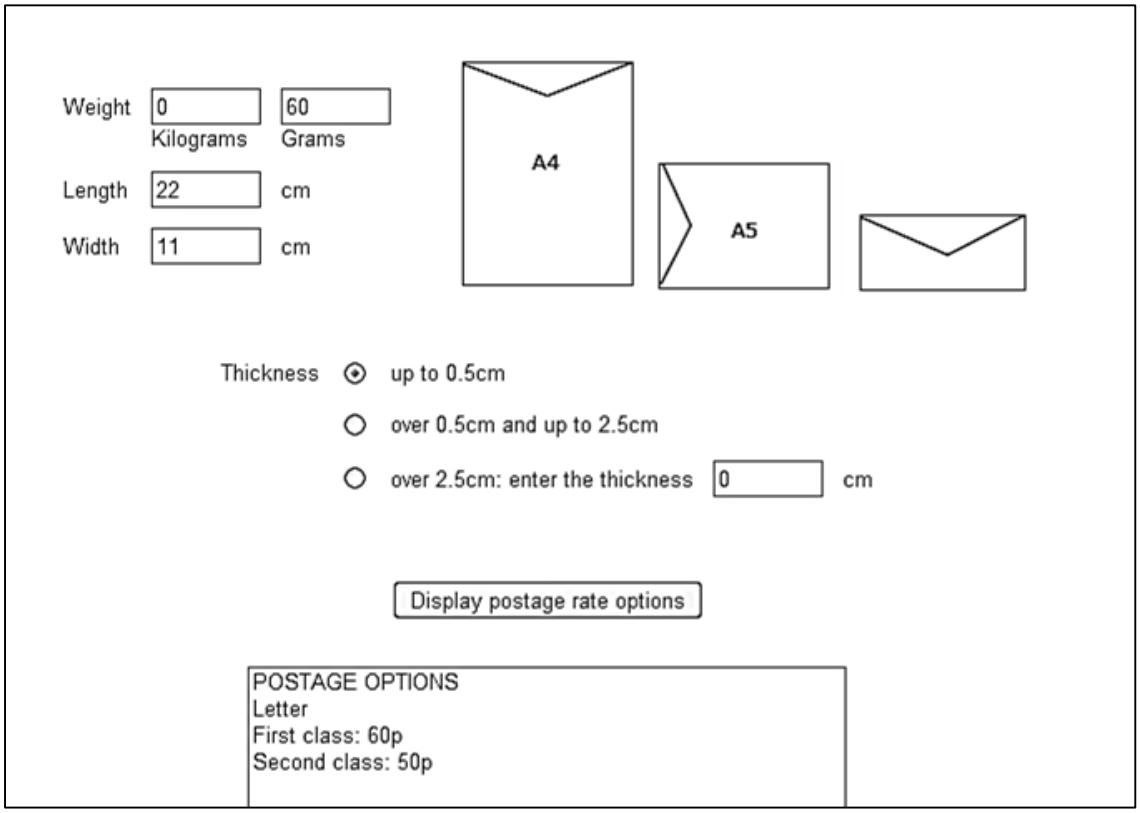
```
    if (letter == false)
    {
        largeLetter = true;
    }
}

if (letter == false && largeLetter == false)
{

    if (length <= 45 && width <= 35 && thickness <= 8 && weight <= 2)
    {
        smallParcel = true;
    }

    if (length <= 61 && width <= 46 && thickness <= 46 && weight <= 20)
    {
        if (smallParcel == false)
        {
            mediumParcel = true;
        }
    }

    if ((weight > 1 && weight <= 30) &&
                    (length > 61 || width > 46 || thickness > 41))
    {
        largeParcel = true;
    }
}
```

We have now identified the postal item as belonging to one of the postage rate categories. The final step is to determine the postage payable.

Begin by adding code to determine and output the first class and second class postage rates in the case of a ***Letter***.

```
        {
            largeParcel = true;
        }
    }

    if (letter == true)
    {
        ListBox1.Items.Add("Letter");
        ListBox1.Items.Add("First class: 60p");
        ListBox1.Items.Add("Second class: 50p");
    }
```

Compile and run the web page. Enter test data for a Letter size and weight then click the ***Display postage rate options*** button. Check that the correct charges are shown.

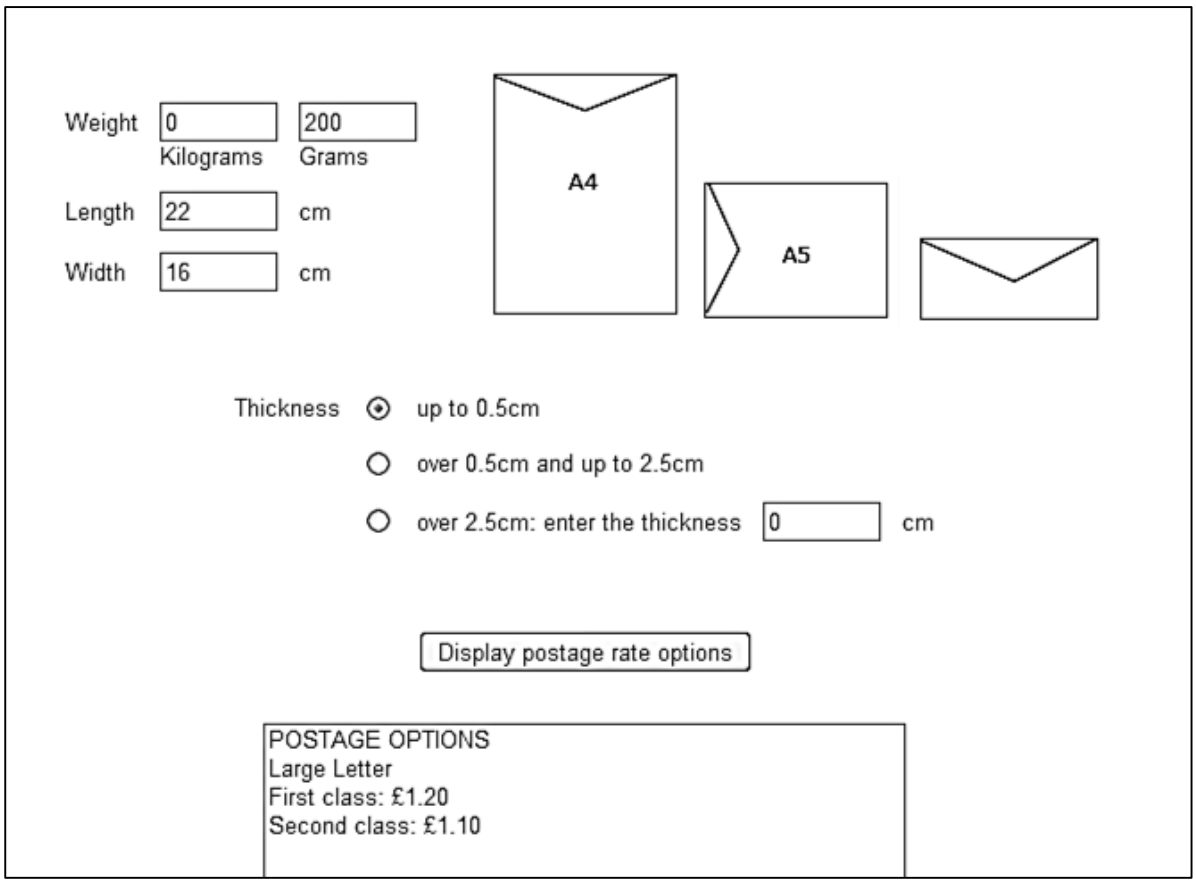


The charges for a ***Large letter*** are more complicated to calculate, as these depend on the weight of the item.

A series of weight bands are given in the postage rates table, and we use ***IF..ELSE..*** conditional structures to find the correct band for the item. Add code to do this.

```
if (letter == true)
{
    ListBox1.Items.Add("Letter");
    ListBox1.Items.Add("First class: 60p");
    ListBox1.Items.Add("Second class: 50p");
}
if (largeLetter == true)
{
    string first;
    string second;
    ListBox1.Items.Add("Large Letter");
    if (weight <= 0.1)
    {
        first = "0.90";
        second = "0.69";
    }
    else
    {
        if (weight <= 0.25)
        {
            first = "1.20";
            second = "1.10";
        }
        else
        {
            if (weight <= 0.5)
            {
                first = "1.60";
                second = "1.40";
            }
            else
            {
                first = "2.30";
                second = "1.90";
            }
        }
    }
    ListBox1.Items.Add("First class: £" + first);
    ListBox1.Items.Add("Second class: £" + second);
}
```

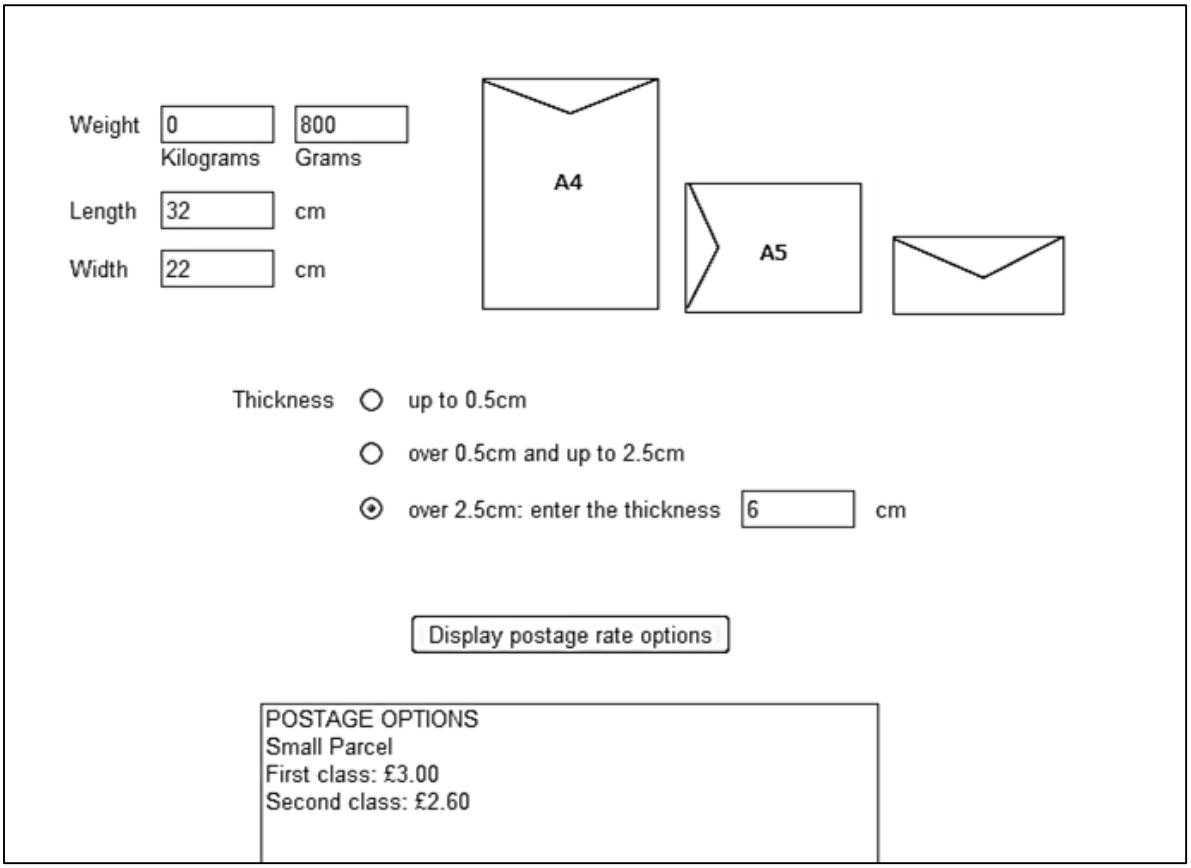
Compile and run the program to check that postage costs for ***Large letters*** are calculated correctly.



The next category to include is ***Small parcels***. We again have more than one weight band. Add the block of code for ***Small parcels*** below the code for ***Large letters***:

```
if (smallParcel == true)
{
    string first;
    string second;
    ListBox1.Items.Add("Small Parcel");
    if (weight <= 1)
    {
       first = "3.00";
       second = "2.60";
    }
    else
    {
       first = "6.85";
       second = "5.60";
    }
    ListBox1.Items.Add("First class: £" + first);
    ListBox1.Items.Add("Second class: £" + second);
}
```

Compile and run the program. Check that different weights of ***Small parcel*** are identified correctly and the corresponding postal charges are displayed.



The postage rate calculations for ***Medium parcels*** and ***Large parcels*** are left as a challenge for you to complete yourself.

In both cases, a number of weight bands will be found in the postage rates table.