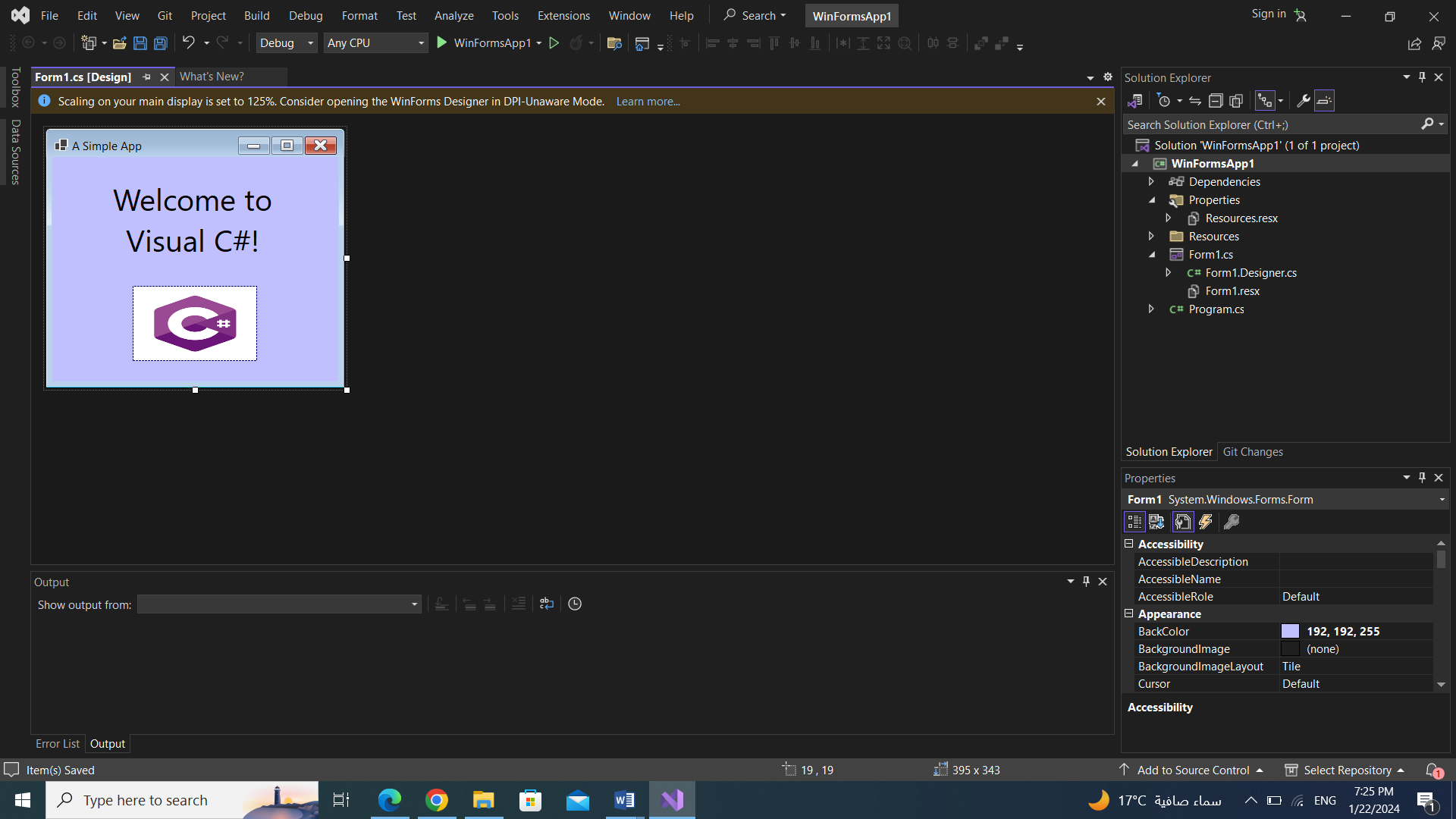
**Sheet (1) - Solution**

**Objective Skills:**

* Design and Implement a simple form.
* Write a code for a simple console application.
* Get to know the best practices for efficient coding.

**Example (1):**

Create a form with the following design using C# windows form application.



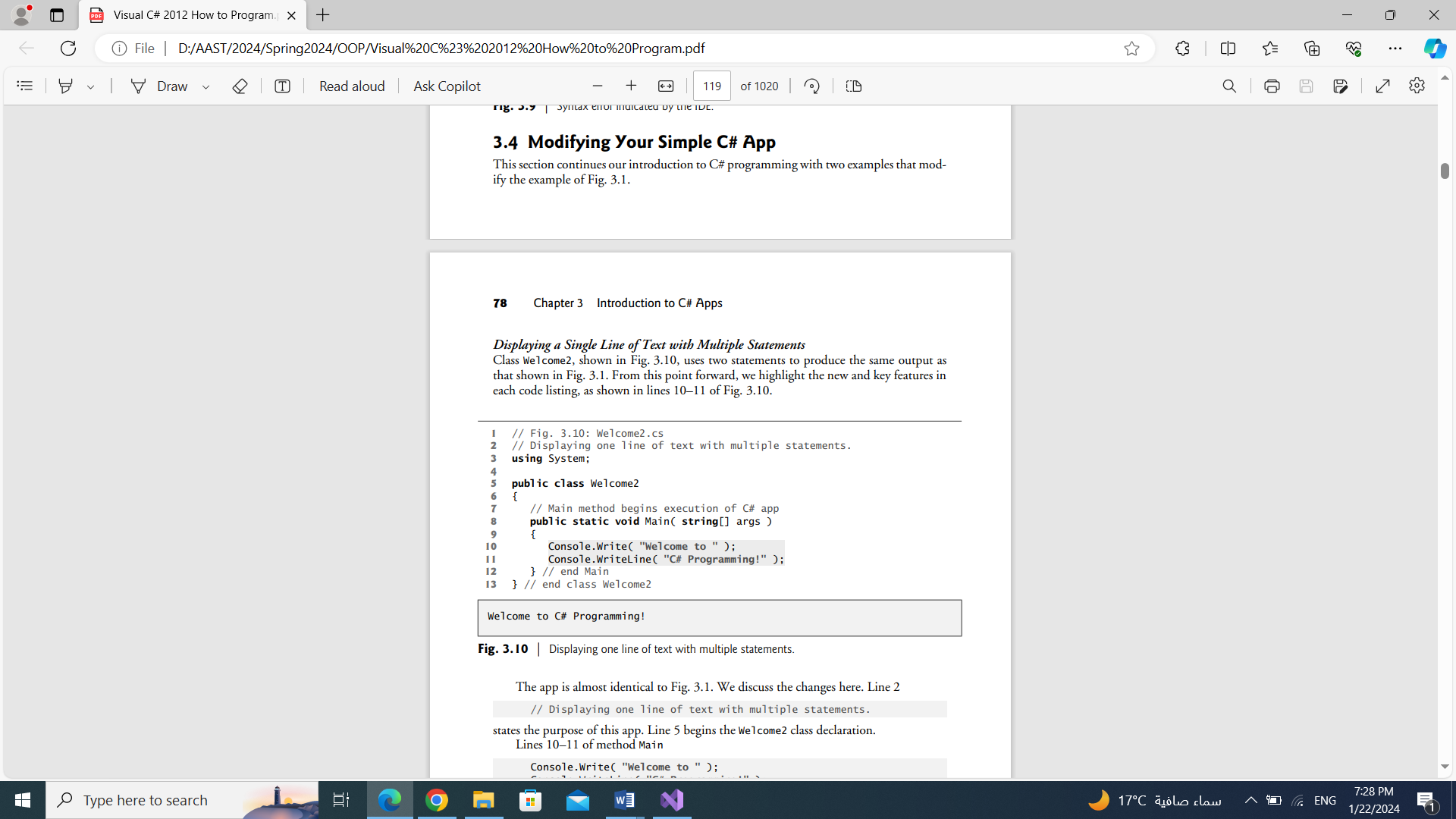
**Solution:**

1. **Creating the new project. To create a new Windows Forms app, select FILE > New Project Select Windows Forms Application. Name the project ASimpleApp, specify the Location where you want to save it (we used the default location) and click OK.**
2. **Setting the text in the Form’s title bar. The text in the Form’s title bar is determined**

**by the Form’s Text property. If the Properties window is not open, select VIEW > Properties Window. Click anywhere in the Form to display the Form’s properties in the Properties window. In the textbox to the right of the Text property, type "A Simple App".**

1. **Resizing the Form. Click and drag one of the Form’s enabled sizing handles (the small white squares that appear around the Form). Using the mouse, select the bottom-right sizing handle and drag it down and to the right to make the Form larger.**
2. **Changing the Form’s background color. The BackColor property specifies a Form’s or control’s background color. Clicking BackColor in the Properties window causes a down arrow button to appear next to the value of the property). When clicked, the down-arrow button displays other options, which vary depending on the property. In this case, the arrow displays tabs for Custom, Web and System (the default). Click the Custom tab to display the palette (a grid of colors). Select the box that represents light blue. Once you select the color, the palette closes and the Form’s background color changes to light blue**
3. **Adding a Label control to the Form. If the Toolbox is not already open, select VIEW > Toolbox to display the set of controls you’ll use for creating your apps. For the type of app. we’re creating in this chapter, the typical controls we use are located in either the All Windows Forms group of the Toolbox or the Common Controls group. If either group name is collapsed, expand it by clicking the arrow to the left of the group name. Next, double click the Label control in the Toolbox. This action causes a Label to appear in the upper-left corner of the. Although double clicking any Toolbox control places the control on the Form, you also can “drag” controls from the Toolbox to the Form.**
4. **Customizing the Label’s appearance. Select the Label by clicking it. Its properties now appear in the Properties window. The Label’s Text property determines the text (if any) that the Label displays. The Form and Label each have their own Text property—Forms and controls can have the same property names (such as BackColor, Text, etc.) without conflict. Set the Label’s Text property to Welcome to Visual C#!. The Label resizes to fit all the typed text on one line. By default, the AutoSize property of the Label is set to True, which allows the Label to update its size to fit all of the text if necessary. Set the AutoSize property to False so that you can resize the Label on your own. Resize the Label (using the sizing handles) so that the text fits. Move the Label to the top center of the Form by dragging it or by using the keyboard’s left and right arrow keys to adjust its position. Alternatively, when the Label is selected, you can center the Label control horizontally by selecting FORMAT > Center in Form > Horizontally.**
5. **Setting the Label’s font size. To change the font type and appearance of the Label’s text, select the value of the Font property, which causes an ellipsis button to appear next to the value. When the ellipsis button is clicked, a dialog that provides additional values, in this case, the Font dialog is displayed. You can select the font name (the font options may be different, depending on your system), font style (Regular, Italic, Bold, etc.) and font size (16, 18, 20, etc.) in this dialog. The Sample text shows the selected font settings. Under Font, select Segoe UI, Microsoft’s recommended font for user interfaces. Under Size, select 24 points and click OK. If the Label’s text does not fit on a single line, it wraps to the next line. Resize the Label so that the words "Welcome to" appear on the Label’s first line and the words "Visual C#!" appear on the second line. Re-center the Label horizontally.**
6. **Aligning the Label’s text. Select the Label’s TextAlign property, which determines how the text is aligned within the Label. A three-by-three grid of buttons representing alignment choices is displayed. The position of each button corresponds to where the text appears in the Label. For this app, set the TextAlign property to MiddleCenter in the three-by-three grid—this selection centers the text horizontally and vertically within the Label. The other Text- Align values, such as TopLeft, TopRight, and BottomCenter, can be used to position the text anywhere within a Label. Certain alignment values may require that you resize the Label to fit the text better.**
7. **Adding a PictureBox to the Form. The PictureBox control displays images. The process involved in this step is similar to that of Step 6, in which we added a Label to the Form. Locate the PictureBox in the Toolbox and double click it to add it to the Form. When the PictureBox appears, move it underneath the Label, either by dragging it or by using the arrow keys.**
8. **Inserting an image. Click the PictureBox to display its properties in the Properties window. Locate and select the Image property, which displays a preview of the selected image or (none) if no image is selected. Click the ellipsis button to display the Select Resource dialog, which is used to import files, such as images, for use in an app. Click the Import… button to browse for an image to insert, select the image file and click OK. To scale the image to the Picture- Box’s size, change the SizeMode property to StretchImage. Resize the PictureBox, making it larger.**
9. **Saving the project. Select FILE > Save All to save the entire solution.**
10. **Running the project. Recall that up to this point we have been working in the IDE design mode (that is, the app being created is not executing). In run mode, the app is executing, and you can interact with only a few IDE features. The text Form1.cs [Design] in the project tab means that we’re designing the Form visually rather than programmatically. If we had been writing code, the tab would have contained only the text Form1.cs. If there is an asterisk (\*) at the end of the text in the tab, the file has been changed and should be saved. Select DEBUG > Start Debugging to execute the app (or you can press the F5 key).**

**Example (2):**

1. Output the following statements using C# console application:
2. 

**Solution:**

**public class Welcome1**

**{**

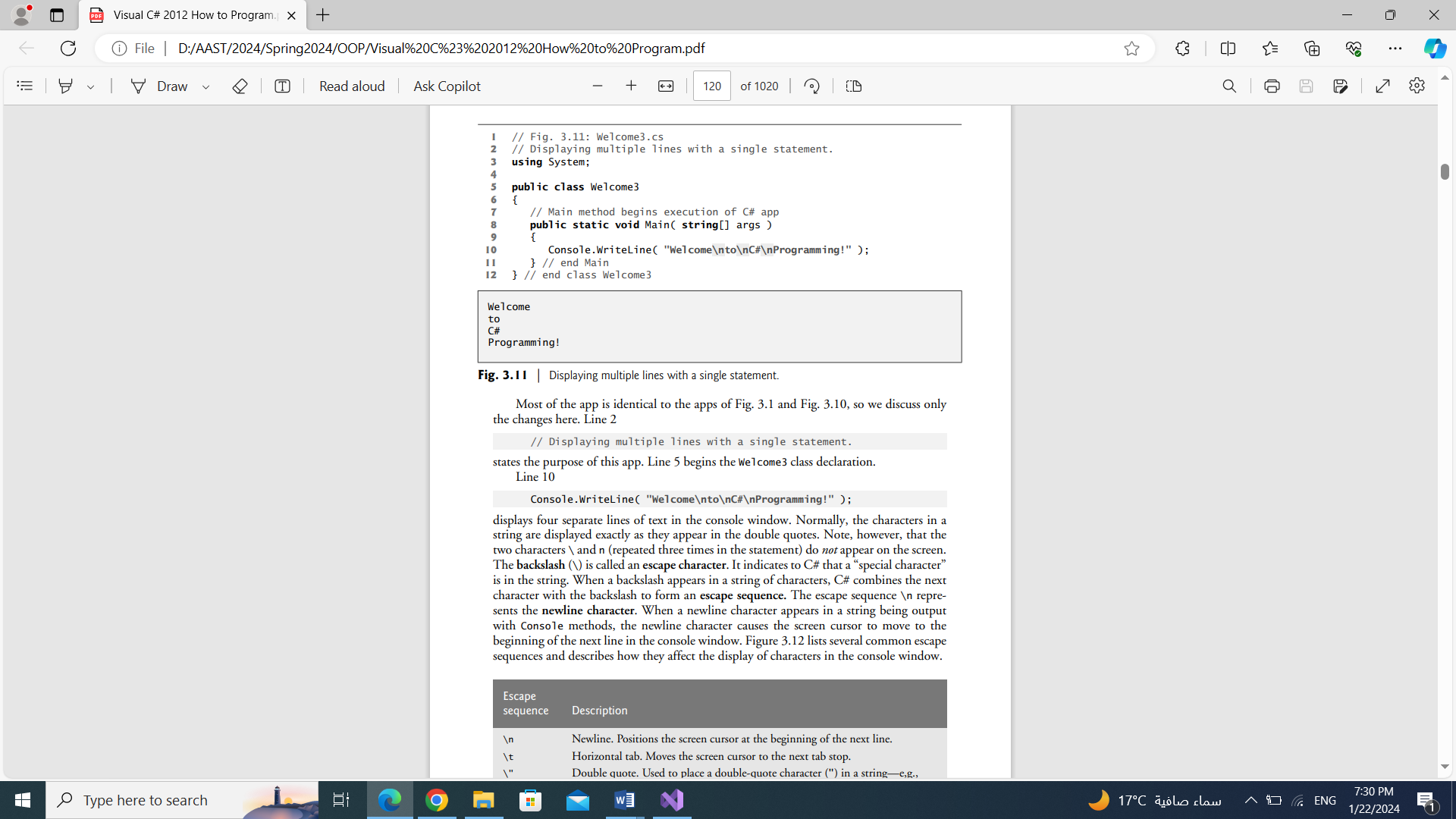
**public static void Main(string[] args)**

**{**

**Console.WriteLine( " Welcome to C# Programming!" );**

**}**

**}**

1. 

**Solution:**

**public class Welcome1**

**{**

**public static void Main(string[] args)**

**{**

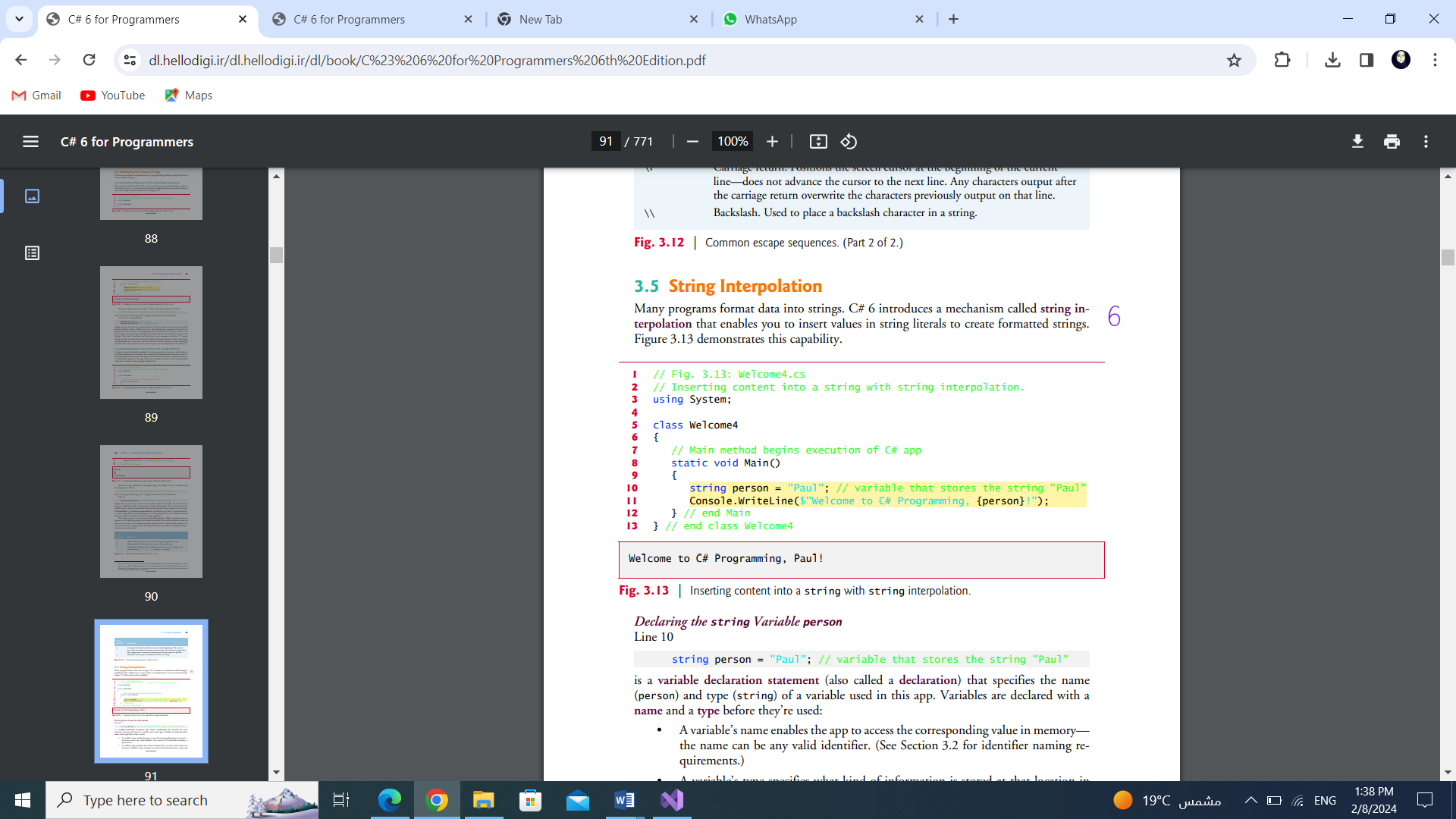
**Console.WriteLine( "Welcome\n to\n C#\n Programming!" );**

**//Or**

**Console.WriteLine( "{0}\n{1}\n{2}\n{3}", "Welcome",”to” "C# “,”Programming!" );**

**}**

**}**



**Solution:**

**public class Welcome1**

**{**

**public static void Main(string[] args)**

**{**

**string person = "Paul**

**Console.WriteLine($"Welcome to C# Programming, {person}!");**

**}**

**}**

**Example (3):**

Write a program that reads two inputs typed by a user at the keyboard, computes the sum of the values and displays the result.

**Solution:**

**public class Welcome1**

**{**

**public static void Main(string[] args)**

**{**

**int number1;**

**int number2;**

**int sum;**

**Console.Write("Enter the first Integer:");**

**number1 = int.Parse(Console.ReadLine());**

**Console.Write("Enter the second Integer:");**

**number2 = int.Parse(Console.ReadLine());**

**sum = number1 + number2;**

**Console.Write($"Sum :{sum}");**

**}**

**}**