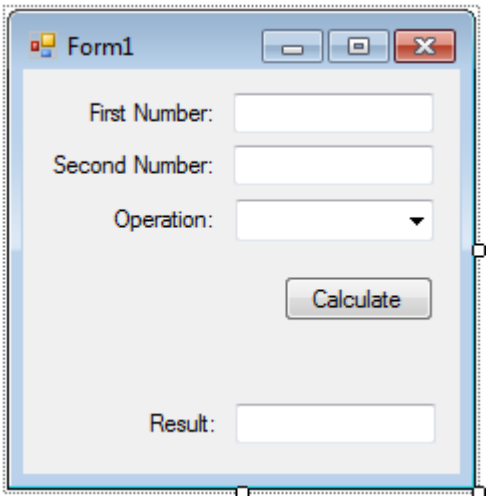
Worksheet I

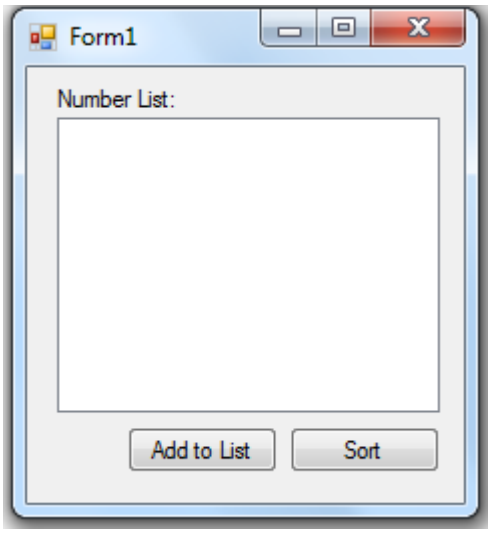
1. Create a windows forms application that adds two numbers. The program gets the two numbers from two text boxes. Finally the result should be displayed in a third text box.
2. Expand the program in #1 in such a way that subtraction, multiplication and division are included. Add a combo box containing the four operations (addition, subtraction, multiplication and division) so that the user selects any operation needed. The interface of the application looks like the following



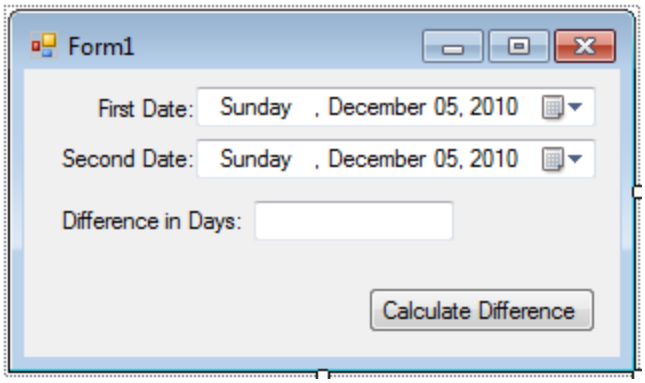
3. Create a program that converts temperature in degree Celsius to Fahrenheit and vice versa. The program will have two text boxes; one to input a value in °C and the other to input a value in °F. The user can give an input in one of the boxes. The program should check what input has been given (°C or °F) and do the corresponding conversion
4. Create a program which calculates the factorial of a number. Use a text box to accept the input number. Display the result using any method you prefer (message box, textbox, or label)
5. Create a program that calculates a tax deduction for a given salary. The tax deduction is progressive and should be according to the following rule:

	Range	**Tax (%)**
	<= 150	0
	150 - 600	10
	600 - 1200	15
	1200 - 2400	20
	2400 - 3500	25
	3500 - 5000	30
	> 5000	35

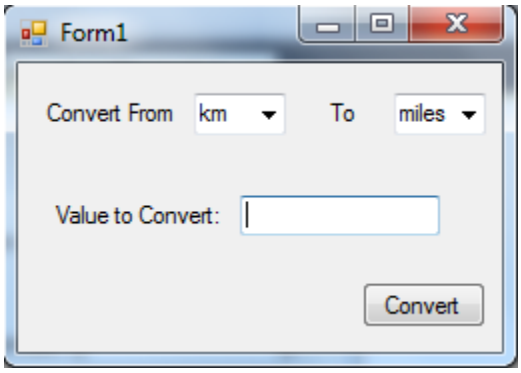
6. Create an ordinary calculator application that performs the four arithmetic operations, square root, reciprocal and memory.
7. Create a program that accepts a list of numbers in random order. The user gives a number one at a time and the number will be added to a list box. Finally, when the user clicks on a sort button the numbers in the list box should be sorted in descending order. The interface of the application should look like the following:



8. Create a program that calculates the number of days between two specific dates. The program should have two date time pickers to set the date range. The result should be displayed using a label or a text box. Look at the interface below:



9. Create a program that converts length from one unit to another. The program should have one text box for input purpose and two combo boxes to select source unit and destination units. Look at the interface below:



10. Expand the above application in such a way that it can also convert weight. The user can select whether to convert length or weight by selecting from a combo box.
11. Assume a class called Shape which represents geometric shapes. This class has an attribute called **shapeName**. Three classes called **Triangle**, **Rectangle**, and **Circle** inherit from the shape class. Triangle has **base** and **height** attributes. Rectangle has **length** and **width** attributes. Circle has **radius** attribute.
    a. Create the above classes using C#
    b. Create an interface which allows the user to select a shape type from a drop down. If the user selects circle, a text box to enter radius should be visible. On the other hand, if the user selects triangle or rectangle, two text boxes for entering base and height, or length and width respectively should be visible. Then when the user clicks on a "Calculate" button, the area of the corresponding shape should be displayed. Alternatively, the user may click another button with a label "Calculate Perimeter" and the perimeter of the selected shape should be calculated.
12. Imagine an application which registers items in a supermarket. Each item belongs to a category. A category has an **Id** and **Name** fields. An item has the fields **Id**, **Name**, **UnitPrice**. In addition to the given fields above, a category also has a field to represent the list of items belonging to it. Similarly, an item has a field representing the category it belongs to.
    a. Create the Category and Item classes per the specification given above.
    b. Assume two forms, one to register categories and the other to register items. Assume also that you save the data in a collection. Write a save event handler for each form.
13. Create a simple function which accepts an integer value and returns an output like the examples shown below:

For an input value of 3: 1
1 3
1 3 5

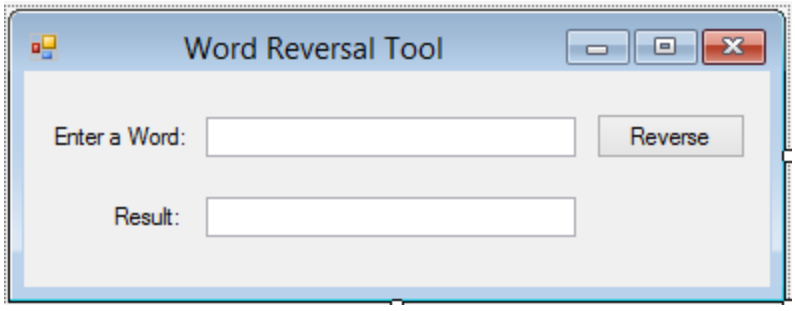
For an input value of 4: 1
1 3
1 3 5
1 3 5 7

For an input value of 5: 1
1 3
1 3 5
1 3 5 7
1 3 5 7 9

And so on.

Use the concept of jagged arrays to solve the problem.

14. Given a list box called **NamesList** which contains a list of names:
    a. Create a function which accepts a name as a parameter and returns true if the name is in the list box or false otherwise.
    b. Create a function which accepts a name as a parameter and returns the index of the name if it exists in the list box, or -1 if it doesn't exist.
15. Consider an application which reverses a given word. The application has a TextBox to accept a word from the user, a Button with the text **Reverse**, and a second TextBox to output the reversed word. See the interface below:



The user is expected to give a word / phrase in the first text box. When the reverse button is clicked the word / phrase will be reversed and the output will be displayed in the result box. Write the event handler for the Reverse button. (Hint: use the **Stack** collection type to do the reversal)