CMPE 1666- Practice Questions for Lab Exam2

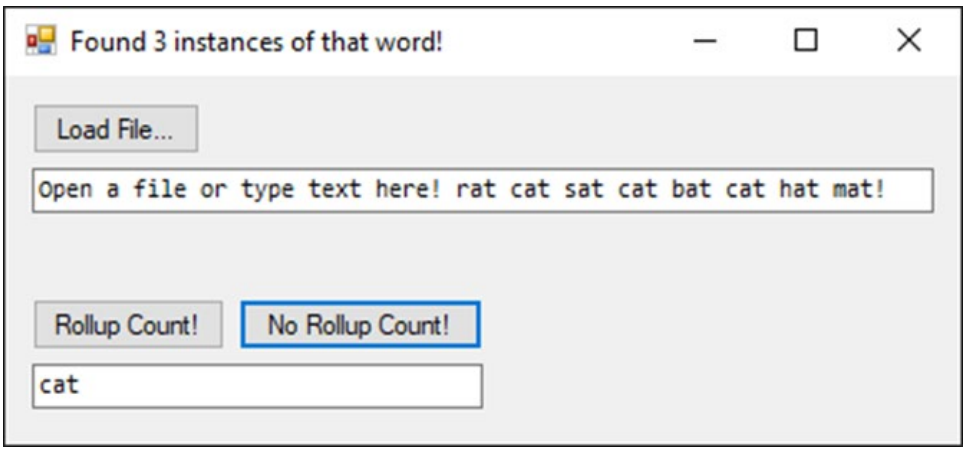
Question 1

This question may be completed one of two ways:

- Using recursion with a roll-up (summing in a return value)
- Using recursion with a global variable (not good practice, worth less marks)

You may apply for check-off using only one form of the recursion.

Create an application with the following appearance:



When the user clicks the 'Load File…' button, they will be presented with an OpenFileDialog that will permit selection of a text file. Select and open the provided text file. The contents of the file will appear in the TextBox below the button. *NOTE: files should only contain a single line of text.*

The user will enter a word to search for in the bottom TextBox.

You will implement only one of the lower buttons. Both buttons will operate the same way, in that they will start a recursive call to count the number of occurrences of a word in the string above.

- The 'Rollup Count!' button will start a recursive method that returns the matching word count from the recursive method.
- The 'No Rollup Count!' button will start a recursive method that alters a class-level (global) variable to count the matching words. This is a terrible practice, but is easier (maybe), so it is worth less marks.

Implement only one button / recursive method.

Display the count of matching words in the Form caption (Text) as shown when the recursive method is complete.

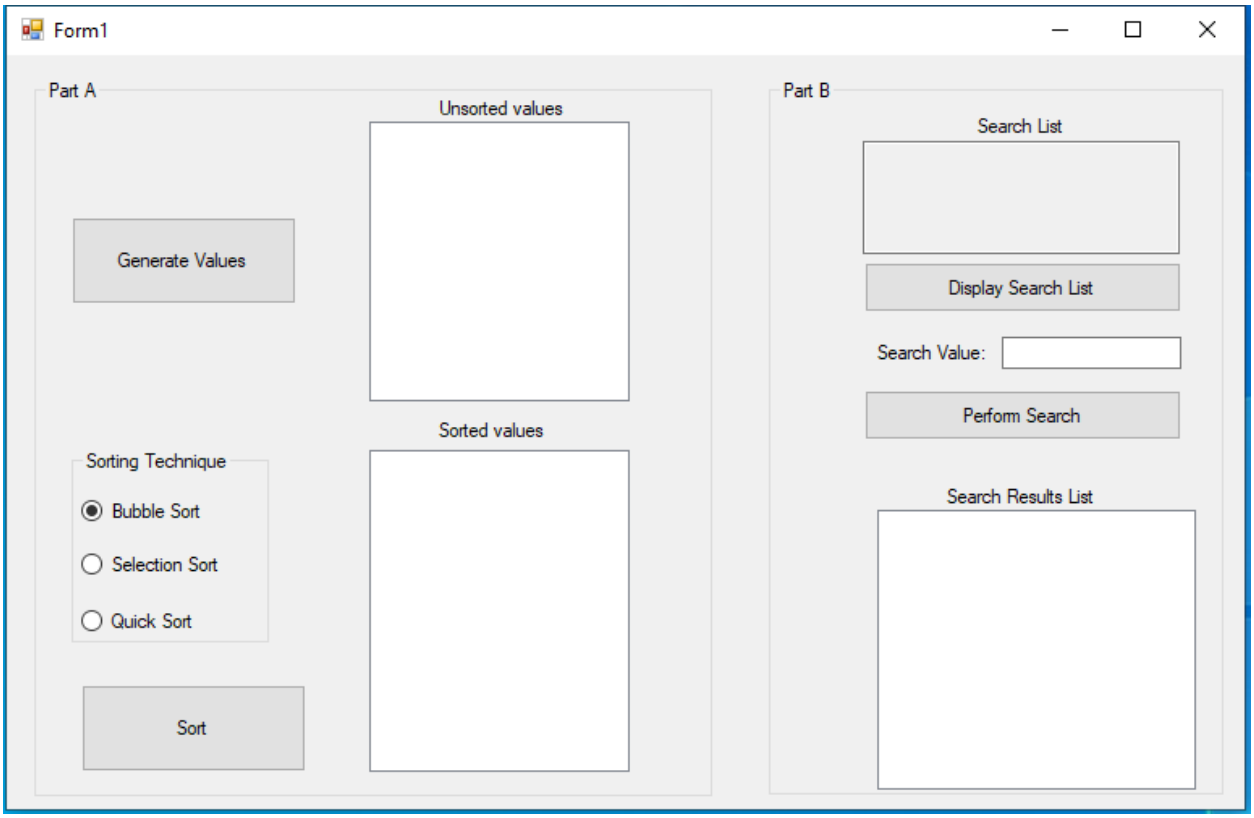
You may use the Split method found in string to split the input string into an array of strings. This would look something like:

```
TextInput.Text.Split(new char[] { ' ' }, StringSplitOptions.RemoveEmptyEntries
```

You may use the ReadAllText method found in File to read the entire contents of a file. The File type is available in the System.IO namespace.

Question 2 – Searching And Sorting

You have been provided with the starter project containing the UI below. The application contains a list of int (called searchList) to be used for part B



Your task is to add methods/event listeners as follows:

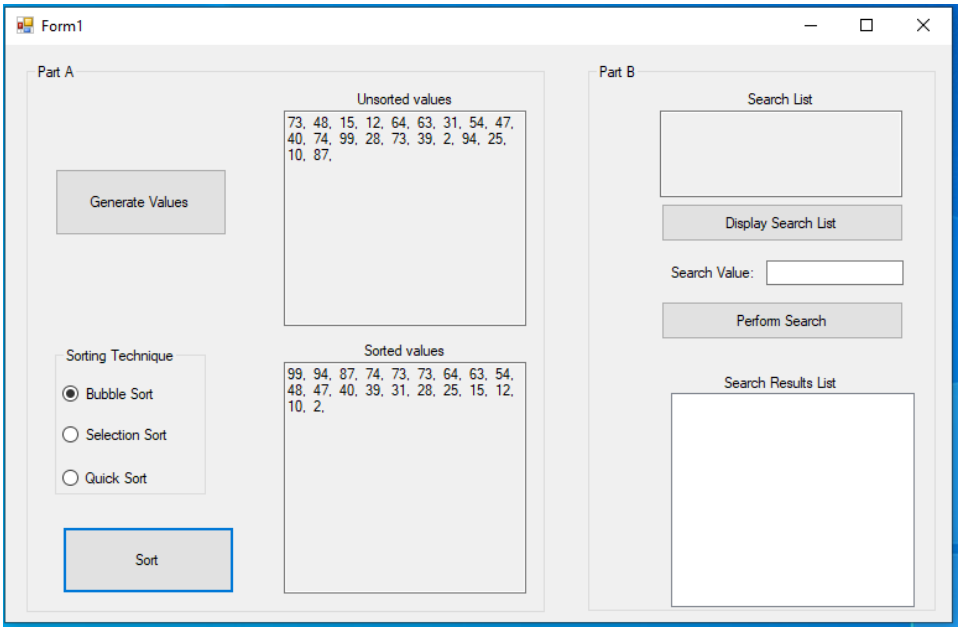
Part A

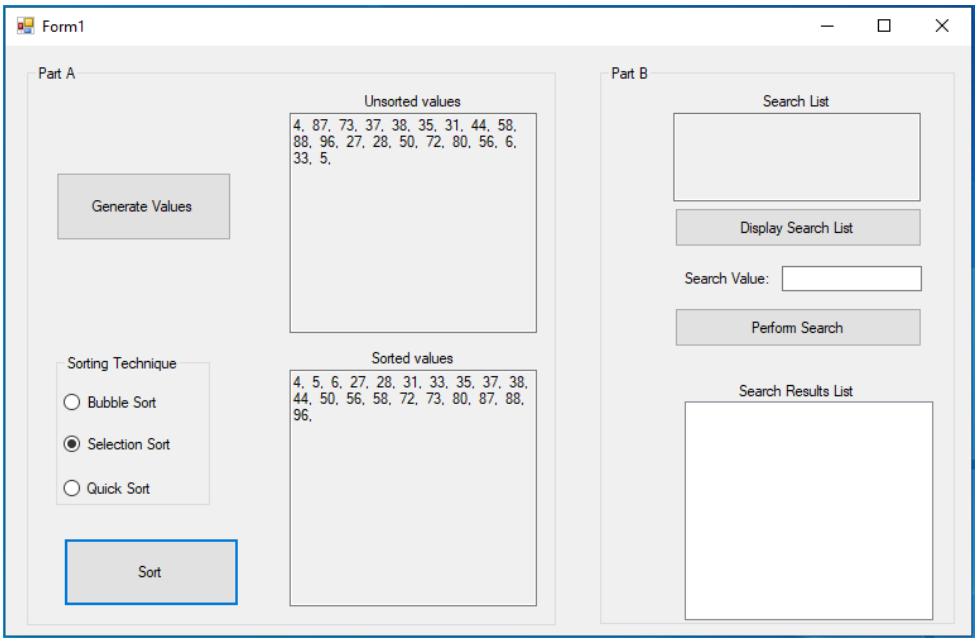
- A click event Listener for the “Generate Values” button so that it generates 20 values of type int in the range of 1..100, adds them to a list and also adds them to the “UnSorted Values” listbox.
- 3 sorting methods that perform sorting of a list of values of type int as below:
    - A method that performs sorting in **descending order** using the Bubble Sort technique
    - A method that performs sorting in **ascending order** using the selection sort technique
    - A method that performs sorting in **ascending order** using the quick sort technique
- A click event listener for the “sort” button, such that it picks one of the 3 sorting methods based on the selected radio button, sorts the data in the generated list and displays the sorted list in the “Sorted Values” list box.

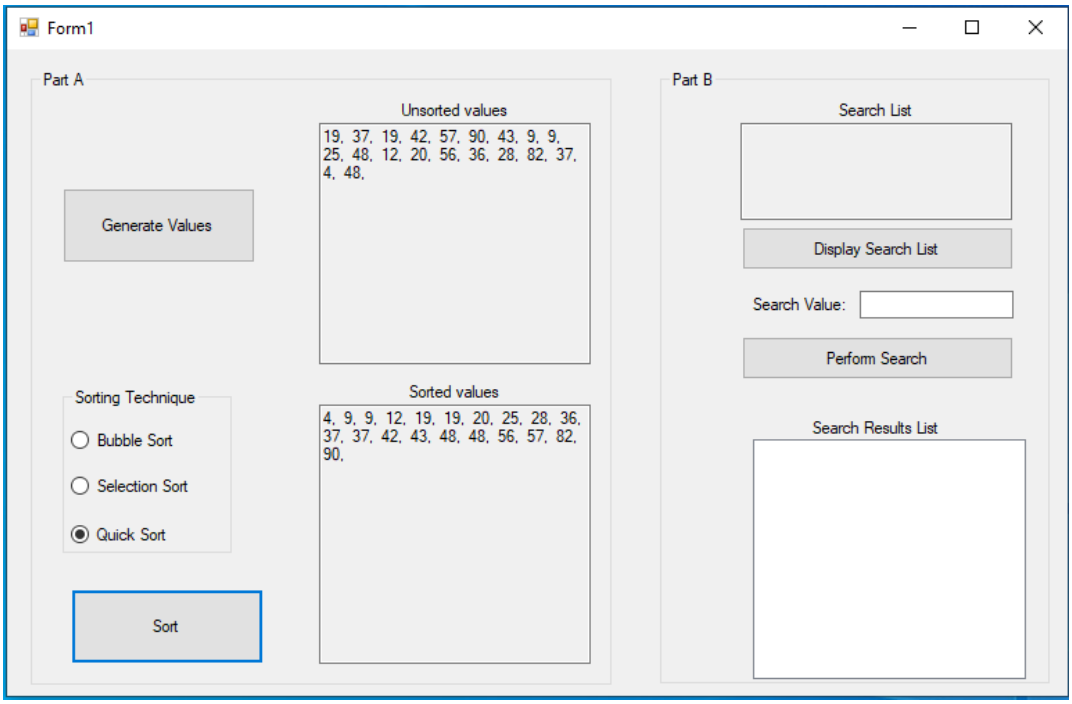
Part B

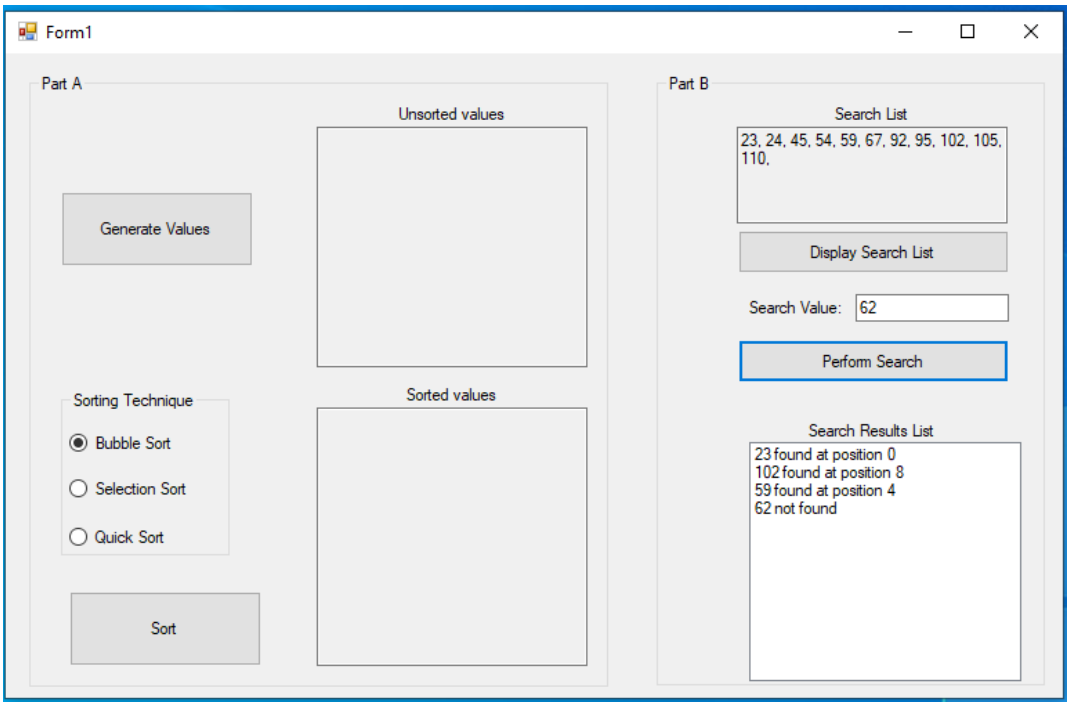
- A click event listener for the “Display Search List” button such that it displays the list searchList (provided) into the multiline textbox.
- A search method using the binary search technique to search for an integer value in an integer list and returns the index where the integer is found. Otherwise, it must return -1.
- A click event listener for the “Perform Search” button such that it obtains the value from the “search value” textbox, searches the value in the search list, and adds the result to the “Search Results List” listbox (as shown in the sample run)

Sample runs:




Form1
Part A
Unsorted values
4, 87, 73, 37, 38, 35, 31, 44, 58, 88, 96, 27, 28, 50, 72, 80, 56, 6, 33, 5,
Generate Values
Sorting Technique
Bubble Sort
Selection Sort
Quick Sort
Sort
Sorted values
4, 5, 6, 27, 28, 31, 33, 35, 37, 38, 44, 50, 56, 58, 72, 73, 80, 87, 88, 96,
Part B
Search List
Display Search List
Search Value:
Perform Search
Search Results List

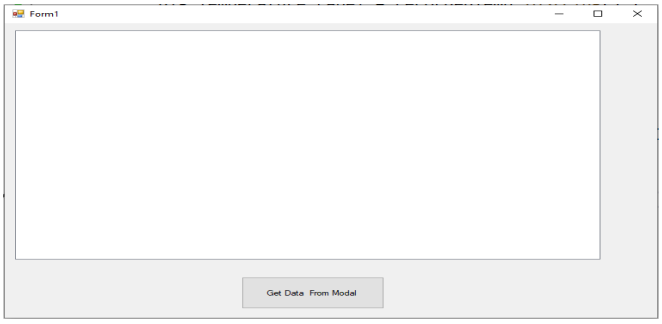

Form1
Part A
Unsorted values
19, 37, 19, 42, 57, 90, 43, 9, 9, 25, 48, 12, 20, 56, 36, 28, 82, 37, 4, 48,
Generate Values
Sorting Technique
Bubble Sort
Selection Sort
Quick Sort
Sort
Sorted values
4, 9, 9, 12, 19, 19, 20, 25, 28, 36, 37, 37, 42, 43, 48, 48, 56, 57, 82, 90,
Part B
Search List
Display Search List
Search Value:
Perform Search
Search Results List

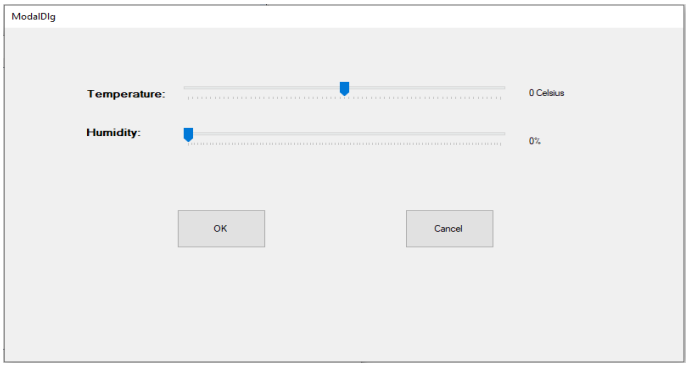


Question 3 - Modal Dialogs

<u>Delegates are not permitted in this question</u>.

1. In this question you will develop a small application to simulate the daily update of 2 items of weather information (namely, temperature and humidity) from a weather station.
2. Create a Form-based application, with the 2 forms and required controls as shown in the figure. The first form is the main (starting) form of the application, while the second one is a Modal Dialog. The Modal Dialog should **not** contain the maximize, minimize and control (X) buttons. It should also be **non-resizable**. All controls must be named appropriately.

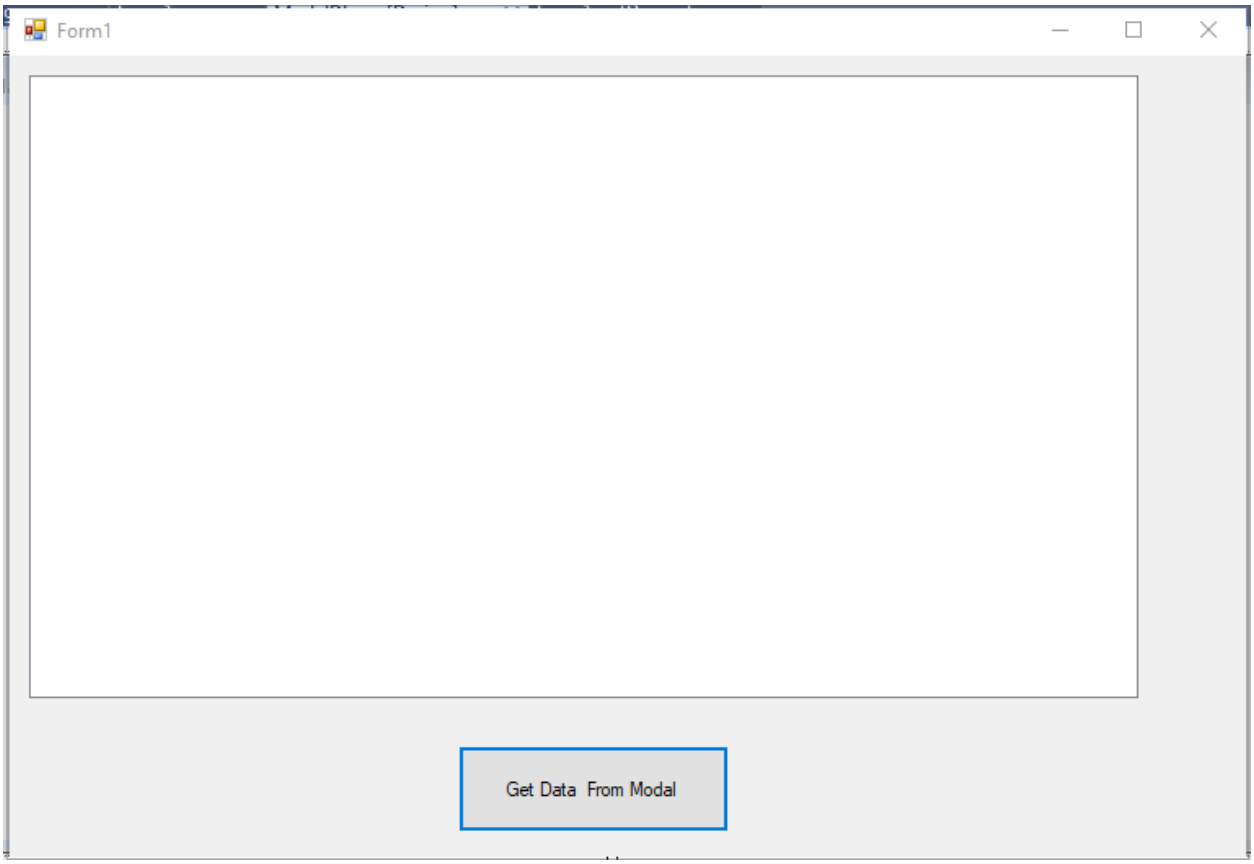




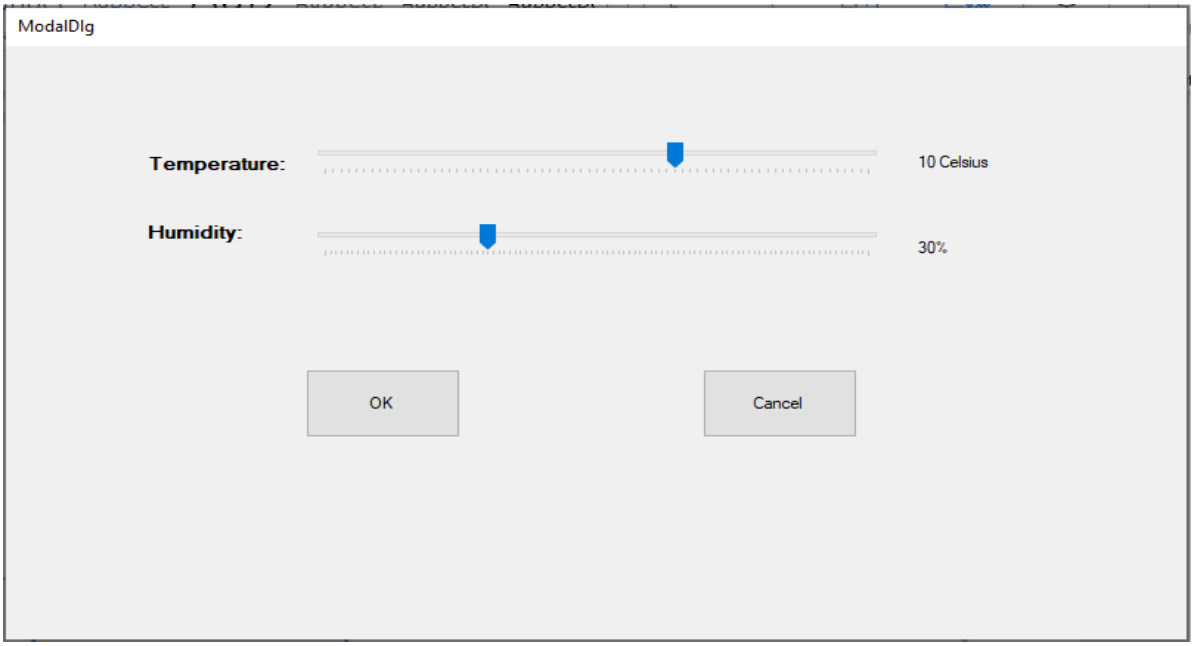
3. The listbox in the main form must grow both horizontally and vertically when the size of the form increases. The "**Get Data From Modal**" button must retain its relative position below the listbox.
4. The main form code must also include the definition of an **enum** type, called **Days**, for all days of the week, with Monday being the first day and value set to 1, and Sunday being the last day.
5. When the application starts running, the main form will appear and the listbox will be empty.
6. When the "**Get Data From Modal**" button is clicked, the modal dialog must appear.
7. The modal dialog allows the user to select a temperature value and humidity value from the respective track bars. The humidity trackbar must be set to cover the range 0-100 in steps of 1. The temperature trackbar must cover the range of -35 to +35 in steps of 1.
8. Each time the user clicks on the "**Get Data from Modal**" to open the Modal Dialog, the main form will consider it to be a new day. We will assume that the first time it opens will be for a Monday. You should ensure that after Sunday, it wraps around to Monday. The days are only used for display in the main form. So they do not need to be communicated to the modal dialog.
9. When the user clicks the **OK** button on the Modal Dialog, the dialog closes. The main form will add the day followed by the values selected on the trackbars (from the modal) to the list box in the format shown (in the sample run).
10. If the user clicks the Cancel button in the modal dialog, the dialog must close but no update will be done in the list box.
11. When the modal dialog is opened for the first time, the value of both the humidity and the temperature must be 0. The user will then select the values by scrolling on the track bars. When the modal dialog is opened subsequently, the trackbars must show the previous values recorded (which will be the last values displayed in the listbox). Even if the user had previously moved the trackbars and chosen Cancel to close the modal dialog (these values won't be recorded), next time it should reopen with the previously **recorded** values. The values in the trackbars must also be displayed in their respective labels and updated as the values on the corresponding trackbar changes.

Note: **Public methods must be created within the modal dialog form code. All data communication between the 2 forms should happen solely through these public methods**.

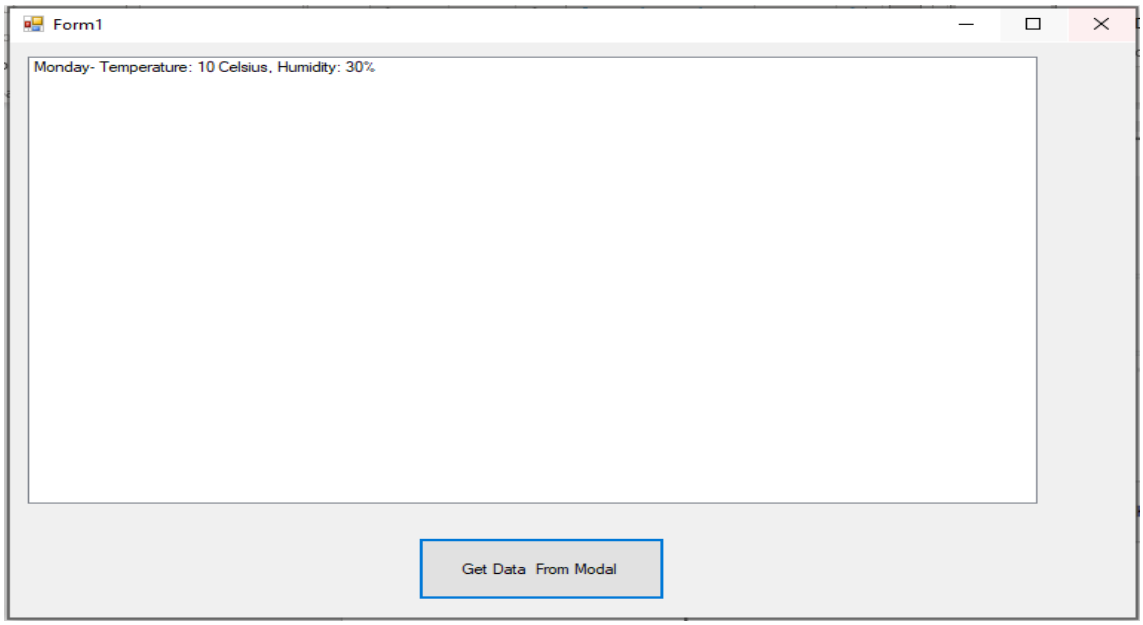
Sample Runs:



On clicking on button modal dialog opens as below:



After values are chosen and we click on OK, results are displayed in the main form as shown below.



As keeps clicking on “Get Data Button” to obtain more data from the modal dialog, the main form will update as below: