**CST8253 Web Programming II**

Lab 3

# Objective

1. Work with C# arrays
2. Create and use C# methods

# Due Date

See Canvas for the due date of this lab. To earn 3 points, you are required:

1. Complete the lab as required.
2. Zip the solution folder (the folder containing **.sln** file) and submit the zipped file to the Blackboard.
3. Demo your lab work during the following week’s lab session.

**Requirements**

Start with the partially completed Visual Studio solution “Lab3Solution”. The class **Program** has a static int array data member initialized to random integers. The class also has several methods declared but not implemented or partially implemented.

You can run included executable “Lab3Solution.exe” to see how your lab 3 solution should work.

1. Implement method **SearchIntArray**

The method **SearchIntArray** takes an unsorted array and a value as input parameter. It loops through all array elements to find out whether the given value is in the array. If found, it returns the index of that value in the array, otherwise it returns -1;

It also “returns” to the caller through a reference argument that the number of comparisons it used to find the index of the given value, or the number of comparisons it used to conclude that the value is not in the array.

1. Implement method **BubbleSort**

From Wikipedia:

“Bubble sort is a simple [sorting algorithm](http://en.wikipedia.org/wiki/Sorting_algorithm) that works by repeatedly stepping through the list to be sorted, comparing each pair of adjacent items and [swapping](http://en.wikipedia.org/wiki/Swap_(computer_science)) them if they are in the wrong order. The pass through the list is repeated until no swaps are needed, which indicates that the list is sorted.”

See <http://en.wikipedia.org/wiki/Bubble_sort> for details and pseudo code of this sorting algorithm.

Do a Google searching, you will find out that there are many implementations (including C#) of bubble search posted on Web. If you cannot figure how to implement the algorithm yourself, you can copy one (put in the comment the URL you copied from) and modify to satisfy the lab requirements.

The method **BubbleSort** takes an array argument as input and sort the elements of the array in ascend order. The method returns the number of “swapping” it performed to sort the array.

1. Implement method **BinarySearch**

From Wikipedia:

A **binary search** or **half-interval search** algorithm finds the position of a specified input value (the search "key") within an array sorted by key value. In each step, the algorithm compares the search key value with the key value of the middle element of the array. If the keys match, then a matching element has been found and its index, or position, is returned. Otherwise, if the search key is less than the middle element's key, then the algorithm repeats its action on the sub-array to the left of the middle element or, if the search key is greater, on the sub-array to the right. If the remaining array to be searched is empty, then the key cannot be found in the array and a special "not found" indication is returned.

See <http://en.wikipedia.org/wiki/Binary_search_algorithm> for details and suggested implementations.

The **BinarySearch** should also “return” to the caller through a reference argument that the number of comparisons it used to find the index of the given value, or the number of comparisons it used to conclude that the value is not in the array.

1. Implement Main( ) method as such (see the sample implementation posted on blackboard with this lab document) :
2. On start, it output the unsorted array elements to the console window.

**Note:** Use the implemented method **PrintArray**.

1. It prompts the user to enter an integer.

For this lab, your application does not have to handle the situation when the user enters non-integers.

1. It calls the method **SearchIntArray** to search the user entered integer in the unsorted array.

If found, it outputs the index of at which the user entered value was found and number of comparisons it used to find the index.

If not found, it outputs the number of comparisons it used to conclude that the user entered value is not in the array.

1. It calls the method **BubbleSort** to sort the array.

After sorting, it outputs the number of swapping it performed to sort the array.

1. It output the sorted array elements to the console window.

**Note:** Use the implemented method **PrintArray**.

1. It prompts the user to enter an integer again.

For this lab, your application does not have to handle the situation when the user enters non-integers.

1. It calls the method **BinarySearch** to search the user entered integer in the sorted array.

If found, it outputs the index of at which the user entered value was found and number of comparisons it used to find the index.

If not found, it outputs the number of comparisons it used to conclude that the user entered value is not in the array.