**Programming Assignment 4**

***Analysis of sorting algorithms***

**Assigned June 25, 2015**

**Due on July 7 2015 (11:59PM on BB)**

## This assignment is worth 50 points

It is well known that for large lists/array divide-and-conquer algorithms, such as *MergeSort* and *QuickSort* are, on average faster than *BubbleSort* and *InsertionSort*. However, for smaller lists, the recursion overhead in *MergeSort* and *QuickSort* wipes out the gain in the divide-and-conquer efficiency.

In this assignment you are to implement each of the four sorting algorithms separately.

Furthermore, you are to implement a *HybridSort*, which in addition to the obvious parameter, the list to be sorted, will take three more, as follows:

1. *Large* -- the name of the algorithm when the list to be sorted is large: *MergeSort*, or *QuickSort*
2. *Small* – the name of the algorithm to be used when the list to be sorted is small: *BubbleSort*, or *InsertionSort*
3. *T* – a threshold, a positive integer such that if the list to be sorted has more then T elements, then *HybridSort* behaves as *Large* sorting algorithm; as soon as the number of elements is less than or equal to T, *HybridSort* switches to *Small* algorithm. Obviously, the list should have at least T elements. You should consider several values of T between 8 and 16.

Thus, a call to *HybridSort* might look like **HybridSort(LIST, “MergeSort”, “InsertionSort”, 6)**, which means that for lists larger than 6 *HybridSort* behaves like *MergeSort* and for lists of 6 or less elements *InsertionSort* is used.

Your task is to:

1. **Basic Implementatio**n. Implement the sorting algorithms described above, each of the classic algorithms, plus *HybridSort*.
   1. Your program should be menu driven and user friendly.
   2. The program should begin by displaying documentation describing the program.
   3. It should ask the user to enter the threshold value
   4. Then the user should ask to enter the list.
      1. The user should first be asked for the size of the list.
      2. For lists of size at most 100, the user should then be asked whether the list is to be entered manually or randomly generated and whether the list is to be displayed
      3. If the user has specified that the list is to be displayed the computer should output the unsorted and sorted lists for all sorting algorithms.
      4. Lists of size greater than 100 should automatically be randomly generated and **not displayed**.

* 1. For the threshold and list entered by the user your program will execute each *BubbleSort*, *InsertionSort*, and each combination *MergeSort* with *BubbleSort, MergeSort* with *InsertionSort, QuickSort* with *BubbleSort, QuickSort* with *InsertionSort*.
  2. Include an outer loop to allow the user to repeat this process as often as desired.

1. **Prepare for Analysis**. After you have Part 1 working, add the option to display the number of comparisons performed by each algorithm.
   1. Use a global counter in each algorithm that is incremented each time a comparison is made to keep track of the total number of comparisons performed.
2. **Perform Analysis**. Run your program for N (e.g. N=50) lists and threshold values {8, 9, 10, 11,…., 16} and
   1. Compute the minimum, average, and maximum number of comparisons across the N lists, for each choice of the threshold value.
   2. Write a short discussion ranking the sorting algorithms and the best choice for the threshold T for the HybridSort algorithm
   3. Include this discussion in a Word (of pdf) file called **discussion**.