LAB S12 – SEARCH AND SORT

You need to be familiar with four sorting algorithms and two searching strategies. In this lab, you will use the provided SortDriver class. This class contains all the methods that you need to write along with helper methods to make them function properly. Each sorting method should count the number of steps required to complete the sorting process. The methods that you need to complete are:

1) sort1 – This will sort the values using the **selection sort** algorithm
2) sort2 – This will sort the values using the **insertion sort** algorithm
3) merge – This method works in conjunction with the mergeSort method. The merge method takes a list with two sorted halves (one half from *first* to *mid*; the second half from *mid+1* to *last*) and merges them into one sorted list.
4) mergeSort – The mergeSort method currently works by using bubble sort. Once you have the merge method working, replace the two calls to bubbleSort with calls to mergeSort to make sure that it works recursively. *This is the only necessary modification for the mergeSort method.*
5) sequentialSearch – This method searches through a list of data sequentially
6) binarySearch – This method uses a binary search algorithm to search through a list

Criteria

1) Each sort method should count the number of steps required to complete the sorting process. Steps should be counted for: assignment statements, comparisons, and method calls.

Extension

If you would like to take this lab further, you can create code that will test and compare the steps required for each sorting algorithm to determine which method is more efficient for various list sizes and arrangements.

Submission

1) Submit your code for this lab on a Googly Doc through Canvas before 11:59 pm on Monday, March 30. Alternatively, you may show your lab to Holm via Google Meet before the due date. ***If you submit through Canvas, you must provide a sample output.***