# Lab 5

1. Determine whether InsertionSort, BubbleSort, SelectionSort are *stable* sorting algorithms, and in each case, give an informal proof of your answer.

2. Perform the MergeSort algorithm by hand on the array [7, 6, 5, 4, 3, 2, 1]. Show all steps, in the way that was done in the lecture.

3. Sometimes MergeSort is supplemented with a secondary sorting routine (typically, InsertionSort is used) in the following way: During the recursion in MergeSort, the size of the array being sorted becomes smaller and smaller. To create a hybrid sorting routine, when a recursive call requires the algorithm to process an array with 20 or fewer elements, give this array to InsertionSort and patch in the result after it has finished. Call this hybrid algorithm MergeSortPlus.

   A. Express the steps of MergeSortPlus in the pseudo-code language we are using in class.
   B. Write the Java code for MergeSortPlus (use the implementation of MergeSort provided in the lab folder)
   C. Run tests to compare running times of MergeSort and MergeSortPlus. Which one runs faster? Explain how you tested and whether you feel your results are conclusive.

4. *Slow Binary Search.* Consider the following algorithm:

   **Algorithm** slowBinSearch(A, x, lower, upper)
       **Input**: Already sorted array A, value x to be
            searched for in array section A[lower]..A[upper]
       **Output**: true or false

     **if** lower > upper **then return** false
     mid ← (upper + lower)/2
     **if** x = A[mid] **then return** true
     **if** getRandomInteger() % 2 = 0 **then**
       **if** x < A[mid] **then**
         **return** binSearch(A, x, lower, mid -1)
       **else**
         **return** binSearch(A, x, mid + 1, upper)
     **else** //if random integer is odd, perform a wasted self-call
       **return** binSearch(A, x, lower, upper)

   Use the technique of counting self-calls to compute the worst-case and average-case asymptotic running times of this algorithm.

5. *Binary Trees.* A *binary tree* is a tree in which every node has at most two children.

   a. Write out 4 different binary trees, each having height = 3 – make sure that no two of your trees have the same number of nodes. (There is no need to give labels to the nodes.)

   b. Examine the trees you have drawn and decide whether the following statement is true or false:

   *Every binary tree of height 3 has at most $2^3=8$ leaves.*

   c. Based on your answer to b, what do you think is true in general about the number of leaves of a binary tree of height $n$?

6. *Self-Aware Arrays.* Say that an integer array int[] arr is *self-aware* if for each $i <$ arr.length, arr[$i$] is the exact number of occurrences of $i$ in arr. For example, [2, 0, 2, 0] is self-aware.

   a. Write an algorithm (and then implement) that accepts as input an array of integers and outputs true if the input array is self-aware. What is the asymptotic running time of your algorithm?

   b. Write an algorithm (and then implement) that accepts as input a positive integer $n$ and, when executed, prints out a list of all self-aware arrays of length $n$. What is the asymptotic running time of your algorithm?