COMP 550.001 - Fall 2017
Assignment 1

Part 1a due: Wednesday, August 30, 2017 (start of class)
Part 1b due: Friday, September 1, 2017 (4:00 p.m.)

For part 1a, you should submit a physical copy of your written homework at the start of class. For part 1b, you should submit a .tar.gz or .zip file with your solutions on Sakai.

Part A

[15 points] Problem 1

Consider the following pseudocode. Suppose that n is an even integer and that $A[1..n]$ is an array whose elements are either α or β, where $\alpha > \beta$.

```
sum = 0
for i = n downto 1 by 2:
    for j = i to n:
        if A[i] ≥ A[j]:
            sum = sum + 1
```

1(a)

As a function of n, what is the *maximum* possible resulting value of the variable *sum*? What is the pattern of entries that leads to this worst case? Express your answer as a summation, and then express the solution to this summation as an exact (not asymptotic) formula involving n. Then express it as an asymptotic formula involving n.

1(b)

As a function of n, what is the *minimum* possible resulting value of the variable *sum*? What is the pattern of entries that leads to this best case? Express your answer as a summation, and then express the solution to this summation as an exact (not asymptotic) formula involving n. Then express it as an asymptotic formula involving n.

[25 points] Problem 2: CLRS Problem 2-2

Bubblesort is a popular, but inefficient, sorting algorithm. It works by repeatedly swapping adjacent elements that are out of order.

```
BUBBLESORT(A)
for i = 1 to A.length − 1:
    for j = A.length downto i + 1:
        if A[j] < A[j − 1]:
            exchange A[j] with A[j − 1]
```

2(a)

Let A' denote the output of BUBBLESORT(A). To prove that BUBBLESORT is correct, we need to prove that it terminates and that

$$A'[1] \leq A'[2] \leq ... \leq A'[n] \tag{1}$$

where $n = A.length$. In order to show that BUBBLESORT actually sorts, what else do we need to prove?

The next two parts will prove inequality (1).

2(b)

State precisely a loop invariant for the **for** loop in lines 2-4, and prove that this loop invariant holds. Your proof should use the structure of the loop invariant proof presented in lecture for Insertion Sort and in Chapter 2.

2(c)

Using the termination condition of the loop invariant proved in part (b), state a loop invariant for the **for** loop in lines 1-4 that will allow you to prove inequality (1). Your proof should use the structure of the loop invariant proof presented in lecture for Insertion Sort and in Chapter 2.

2(d)

What is the worst-case running time of Bubblesort? How does it compare to the running time of Insertion Sort?

[20 points] Problem 3: Subset of CLRS Problem 3-3(a)

Rank the following functions by order of growth; that is, find an arrangement $g_1, g_2, ..., g_{16}$ of the functions satisfying $g_1 = \Omega(g_2), g_2 = \Omega(g_3), ..., g_{15} = \Omega(g_{16})$. Partition your list into equivalence classes such that functions $f(n)$ and $g(n)$ are in the same class if and only if $f(n) = \Theta(g(n))$.

If you want partial credit, be sure to include your reasoning for each relation between and within equivalence classes.

2^{2^n}	$n \lg n$	n^2	$n!$
$\lg n$	n^3	$\lg^2 n$	2^n
$\ln \ln n$	$\lg^* n$	n	$\ln n$
$2^{\lg n}$	1	$2^{2^{n+1}}$	$4^{\lg n}$

Part B

[20 points] Problem 1

For this problem, you are given the implementations of four algorithms discussed in the book. Your goal is to compare their runtimes on a variety of different inputs.

Although you can work with another student for the part B problems, you should generate the results for this problem on your own computer. Make sure to add a `readme.txt` file stating either that you worked alone, or the name of your partner.

1(a)

Give the machine specs for the machine on which you're running the comparison in this problem. For example: 64-bit Windows 7 Ultimate Service Pack 1, Intel i5-6600K CPU @ 3.50 GHz, 32 GB RAM, Java JRE 1.8.

1(b)

Modify `sortFunctionChoice` and `datasetChoice` in `sort/Main.java` in order to run each algorithm with each input. The algorithm implementations and timing code have been provided for you. You should record the times in a table like the one below. Make sure to record the units correctly!

	Insertion Sort	Merge Sort	Selection Sort	Bubble Sort
Small sorted				
Small almost-sorted				
Small backwards				
Small random				
Large sorted				
Large almost-sorted				
Large backwards				
Large random				

1(c)

Using your knowledge of the sorting algorithms from the book and/or lecture, explain in your own words any trends you see, including for each data set why a given sorting algorithm performed the fastest.

[20 points] Problem 2: variant of CLRS Exercise 4.1-3

This problem compares the brute-force and recursive approaches to solving the maximum-subarray problem.

2(a)

Implement both the brute-force and recursive algorithms for the maximum-subarray problem. You should fill in the implementations for the functions `findMaximumSubarrayBruteForce` and `findMaximumSubarrayRecursive` in `maximumSubarray/Solver.java`. For reference, the brute-force pseudocode is given below. The recursive approach is given in Section 4.1 of CLRS.

```
FIND-MAXIMUM-SUBARRAY-BRUTE-FORCE(A)
bestRange = (1, 1)
bestVal = A[1]
for i = 1 to n:
    currentVal = 0
    for j = i to n:
        currentVal = currentVal + A[j]
        if currentVal ≥ bestVal:
            bestVal = currentVal
            bestRange = (i, j)
return (bestRange.low, bestRange.high, bestVal)
```

2(b)

Run your code on random inputs of different sizes by modifying `numElements` and `functionChoice` in `Main.java`. How does each algorithm scale as the input size grows? You should consider inputs up to 100,000 elements, and present your results in a table like the one below.

Input size	10	100	1000	10,000	100,000
Brute-Force					
Recursive					