CSE 21 Spring 2016
Homework 2
Due: Friday, August 5, 2016 at 11:59pm

Instructions

Homework should be done in groups of **one to two** people. You are free to change group members at any time throughout the quarter. Problems should be solved together, not divided up between partners. A **single representative** of your group should submit your work through Gradescope. Submissions must be received by 11:59pm on the due date, and there are no exceptions to this rule.

Students should consult their textbook, class notes, lecture slides, instructors, TAs, and tutors when they need help with homework. Students should not look for answers to homework problems in other texts or sources, including the internet. Only post about graded homework questions on Piazza if you suspect a typo in the assignment, or if you don't understand what the question is asking you to do. Other questions are best addressed in office hours.

Your assignments in this class will be evaluated not only on the correctness of your answers, but on your ability to present your ideas clearly and logically. You should always explain how you arrived at your conclusions, using mathematically sound reasoning. Whether you use formal proof techniques or write a more informal argument for why something is true, your answers should always be well-supported. Your goal should be to convince the reader that your results and methods are sound.

For questions that require pseudocode, you can follow the same format as the textbook, or you can write pseudocode in your own style, as long as you specify what your notation means. For example, are you using "=" to mean assignment or to check equality? You are welcome to use any algorithm from class as a subroutine in your pseudocode. For example, if you want to sort list A using InsertionSort, you can call InsertionSort(A) instead of writing out the pseudocode for InsertionSort.

Required Reading Rosen Sections 3.1 and 5.5.

Key Concepts Sorting algorithms, including selection (min) sort, insertion sort, and bubble sort; loop invariants and correctness proofs; searching algorithms; counting comparisons.

For the next three problems, refer to the two algorithms below:

```
BubbleSort( A[1, ..., n], array of integers)
1.    For i from 1 To n-1
2.       For j from 1 To n-i
3.          If A[j] > A[j+1] Then
4.             Interchange A[j] and A[j+1]
```

```
RevisedBubbleSort( A[1, ..., n], array of integers)
1.    For i from 1 To n-1
2.       done := true
3.       For j from 1 To n-i
4.          If A[j] > A[j+1] Then
5.             Interchange A[j] and A[j+1]
6.             done := false
7.       If done Then Break
```

1. (a) (2 points) How many comparisons does BubbleSort make on an array of size n that is already sorted from smallest to largest?

   (b) (2 points) How many comparisons does BubbleSort make on an array of size n that is sorted from largest to smallest?

   (c) (2 points) Explain the difference between BubbleSort and RevisedBubbleSort.

   (d) (2 points) How many comparisons does RevisedBubbleSort make on an array of size n that is already sorted from smallest to largest?

   (e) (2 points) How many comparisons does RevisedBubbleSort make on an array of size n that is sorted from largest to smallest?

2. (10 points) Prove the following loop invariant for BubbleSort.
   After the i^{th} pass, positions $A[n-i+1, \ldots, n]$ contain the i largest elements of the input, in sorted order.

3. (10 points) List all possible arrays containing the numbers 1, 2, 3, 4, and 5 for which RevisedBubbleSort will terminate with a value of $i = 2$.

4. In this problem, we are given a sequence $a_1, a_2, \ldots, a_n$ of integers and we want to return a list of all terms in the sequence that are greater than the sum of all previous terms of the sequence. For example, on an input sequence of $1, 4, 6, 3, 2, 20$, the output should be the list $1, 4, 6, 20$. The following algorithm solves this problem.

   procedure PartialSums($a_1, a_2, \ldots, a_n$: a sequence of integers with $n \geq 1$)

   1. $total := 0$
   2. Initialize an empty list L.
   3. **for** $i := 1$ to n
   4. **if** $a_i > total$
   5. Append a_i to list L.
   6. $total := total + a_i$
   7. **return** L

   (a) (6 points) Prove the following loop invariant by induction on the number of loop iterations:

   Loop Invariant: After the kth iteration of the **for** loop, $total = a_1 + a_2 + \cdots + a_k$ and L contains all elements from $a_1, a_2, \ldots, a_k$ that are greater than the sum of all previous terms of the sequence.

   (b) (4 points) Use the loop invariant to prove that the algorithm is correct, i.e., that it returns a list of all terms in the sequence that are greater than the sum of all previous terms of the sequence.