INSTRUCTIONS

Homework should be done in groups of one to three 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 10:00pm on the due date, and there are no exceptions to this rule.

You will be able to look at your scanned work before submitting it. Please ensure that your submission is legible (neatly written and not too faint) or your homework may not be graded.

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. You may ask questions about the homework in office hours, but not on Piazza.

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 and justify your answers with 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 3.2 and 3.3

KEY CONCEPTS Asymptotic notation, including the definitions of O, Θ, and Ω; best-case, worst-case, average-case; analyzing the run-time of algorithms
1. (7 points) Let \( a(n) \) and \( b(n) \) be functions from the nonnegative integers to the positive real numbers. We say that \( a(n) \in \Omega(b(n)) \) if there exist positive constants \( C \) and \( k \) such that \( a(n) \geq C \cdot b(n) \) for all \( n > k \).

Now let \( f(n), g(n), \) and \( h(n) \) be functions from the nonnegative integers to the positive real numbers. Prove the following transitive property from the above definition of \( \Omega \):

\[
\text{If } f(n) \in \Omega(g(n)) \text{ and } g(n) \in \Omega(h(n)) \text{ then } f(n) \in \Omega(h(n)).
\]

2. For each part, say whether the statement is true or false and justify your answer. All logarithms are base 2 unless otherwise noted.

(a) (4 points) \( 2^n \in \Theta(4^n) \)

(b) (4 points) \( \log(n^2) + \log(10^{10}n^{10}) \in O(\log n) \)

(c) (4 points) \( \sqrt{2^n} \in O((\sqrt{2})^n) \)

(d) (4 points) \( \frac{n}{\ln n} \in \Omega(\ln(n)^2) \)

(e) (4 points) If \( \log f(n) \in \Theta(\log g(n)) \), then \( f(n) \in \Theta(g(n)) \).

3. The following algorithm takes as input a list \( h_1, h_2, \ldots, h_n \) of \( n \) homework scores, where \( n \geq 2 \). Each homework score is a real number from 0 to 100. The algorithm computes the average of the homework scores, where the lowest score is dropped (not included in the calculation.)

\[
\text{procedure HWAvg}(h_1, h_2, \ldots, h_n: \text{a list of } n \geq 2 \text{ homework scores between 0 and 100})
\]

1. \( m := 1 \)
2. \( \text{for } i := 2 \text{ to } n \)
3. \( \quad \text{if } h_i < h_m \text{ then } m := i \)
4. \( \quad \text{total} := 0 \)
5. \( \text{for } j := 1 \text{ to } n \)
6. \( \quad \text{if } j \neq m \text{ then } \text{total} := \text{total} + h_j \)
7. \( \text{return } \text{total}/(n - 1) \)

(a) (3 points) What is the worst-case runtime of the HWAvg algorithm in \( \Theta \) notation? Justify your answer by referring to the pseudocode.

(b) (5 points) Give pseudocode for a different algorithm, SimplerHWAvg, that uses only one loop to solve the same problem, and analyze its worst-case runtime in \( \Theta \) notation. What effect does using only one loop have on the algorithm’s efficiency?
4. Suppose we are given a list of real numbers, and we want to know whether the list contains any duplicate elements. The following two algorithms solve this problem, returning true if some pair of list elements are the same, and false if not.

procedure MatchExistsA(a1, a2, ..., an: a list of real numbers with n ≥ 1)
1. for i := 2 to n
2. for j := 1 to i – 1
3. if aj == ai then
4. return true
5. return false

procedure MatchExistsB(a1, a2, ..., an: a list of real numbers with n ≥ 1)
1. BubbleSort(a1, a2, ..., an)
2. for i := 1 to n – 1
3. if ai == ai+1 then
4. return true
5. return false

(a) (5 points) Describe all worst-case inputs to each algorithm. Find the worst-case running time of each algorithm using Θ notation. Justify your answers by referring to the pseudocode. Which algorithm is more efficient in the worst case?

(b) (5 points) Describe all best-case inputs to each algorithm. Find the best-case running time of each algorithm using Θ notation. Justify your answers by referring to the pseudocode. Which algorithm is more efficient in the best case?

5. (5 points) In this problem, your goal is to identify who among a group of people has a certain disease. You collect a blood sample from each of the people in the group, and label them 1 through n. Suppose that you know in advance that exactly one person is infected with the disease, and you must identify who that person is by performing blood tests. In a single blood test, you can specify any subset of the samples, combine a drop of blood from each of these samples, and immediately get a result. If any sample in the subset is infected, the test will come up positive, otherwise it will come up negative. Your goal is to develop a strategy to find the infected person with as few blood tests as possible.

This idea of testing multiple samples at a time has been used in history at times when it was impractical or expensive to perform blood tests, for example, to find out which soldiers in World War II were carrying diseases. In those situations, the problem was even harder because there could be any number of infected people in the group. Later, we will encounter this same problem in more generality.

Give an algorithm that finds the infected sample in a set of n blood samples, using as few tests as you can. Write pseudocode and a high-level English description of how your algorithm works. You don’t need to prove the correctness of your algorithm or analyze its runtime.