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 11:59pm on the due date, and there are no exceptions to this rule.

Homework solutions should be neatly written or typed and turned in through Gradescope by 11:59pm on the due date. No late homeworks will be accepted for any reason. 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. 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 Chapter 5 and Sections 8.1-8.3.

KEY CONCEPTS Recursive algorithms (definitions, proofs of correctness, runtime calculations), recursive structure of a problem, recurrence relations, guess-and-check method for solving recurrences, unraveling recurrences.
1. In each situation, write a recurrence relation, including base case(s), that describes the recursive structure of the problem. You do not need to solve the recurrence.

(a) (2 points) When you cut a pizza, you cut along a diameter of the pizza. Let $P(n)$ be the number of slices of pizza that exist after you have made $n$ cuts, where $n \geq 1$. Write a recurrence for $P(n)$.

(b) (2 points) A bunch of motorcycles and SUVs want to parallel park on a street. The street can fit $n$ motorcycles, but SUVs take up three motorcycle spaces. Let $A(n)$ be the number of arrangements of cars and motorcycles on a street that fits $n$ motorcycles. For example, $A(5) = 4$ because there are four ways to park vehicles on a street with five motorcycle spaces. If M stands for motorcycle, and C stands for car, then the four arrangements are: MMC, MCM, CMM, and MMMMM. Write a recurrence for $A(n)$.

(c) (2 points) Let $B(n)$ be the number of length $n$ bit sequences that have no three consecutive 0’s. Write a recurrence for $B(n)$.

(d) (2 points) Let $C(n)$ be the number of $1 \times 1$ cells in an $n \times n$ grid. Write a recurrence for $C(n)$.

(e) (2 points) A ternary string is like a binary string except it uses three symbols, 0, 1, and 2. For example, 12210021 is a ternary string of length 8. Let $T(n)$ be the number of ternary strings of length $n$ with the property that there is never a 2 appearing anywhere after a 0. For example, 12120110 has this property but 10120012 does not. Write a recurrence for $T(n)$.

2. (a) (5 points) Suppose a function $g$ is defined by the following recursive formula, where $n$ is a positive integer. $g(n) = 4g(n/2) + n^2$. Use the Master theorem to solve for $g$ up to $O$.

(b) (5 points) Suppose a function $f$ is defined by the following recursive formula, where $n$ is a positive integer. $f(n) = 2f(n/3) + O(n)$. Use the Master theorem to solve for $f$ up to $O$.

3. Present the predecessor algorithm from homework 2 as a recursive algorithm, and prove it is correct. Give a recurrence for the time complexity of this algorithm, and use the master theorem to solve it. (3 points algorithm description, 4 points proof of correctness, 3 points recurrence.)

4. The following algorithm determines whether a word is a palindrome, that is, if the word is the same read left to right as right to left. An example of a palindrome is $racecar$.

    procedure Palindrome($s_1s_2s_3 \ldots s_n$)
    1.      if $n = 0$ or $n = 1$ then return $true$
    2.      if $s_1 = s_n$ then return Palindrome($s_2 \ldots s_{n-1}$)
    3.      else return $false$

    Note: Writing $s_1s_2s_3 \ldots s_n$ denotes a string of length $n$ whose characters are $s_1$, $s_2$, $s_3$, etc. These characters are being concatenated (not multiplied) to form a string.

(a) (6 points) Prove that this algorithm is correct, i.e., that it returns true if and only if $s_1s_2s_3 \ldots s_n$ is a palindrome.

(b) (2 points) Let $C(n)$ be the number of times this algorithm compares two characters $s_i$ and $s_j$ for some $i, j$. Write a recurrence relation that $C(n)$ satisfies.

(c) (2 points) Solve the recurrence found in part (b) and write the solution in $\Theta$ notation.

5. Say we are given two polynomials of degree $n-1$ as their arrays of co-efficients, $a_0, a_{n-1}$ and $b_0, b_{n-1}$, so that $a(x) = \sum_{i=0}^{n-1} a_i x^i$ and $b(x) = \sum_{i=0}^{n-1} b_i x^i$ and we want to compute the description of $a(x)b(x)$, their product.

For all parts, you are not required to prove correctness of your algorithms, but are required to give a time analysis in $O$ form.
(a) Give the straight-forward, iterative algorithm for this problem. (5 points)

(b) Give a simple divide-and-conquer algorithm for this problem. (5 points)

(c) Use the Karatsuba method to give an improved divide-and-conquer algorithm for this problem. (10 points).