The aim of this assignment is to give you some practice analyzing runtime and proving correctness of recursive algorithms.

For each question, please write up detailed answers carefully. Make sure that you use notation and terminology correctly, and that you explain and justify what you are doing. Marks will be deducted for incorrect or ambiguous use of notation and terminology, and for making incorrect, unjustified, ambiguous, or vague claims in your solutions. You should clearly cite any sources or people you consult, other than the course notes, lecture materials, and tutorial exercises.

Your assignment must be typed to produce a PDF document a2.pdf (hand-written submissions are not acceptable). You may work on the assignment in groups of 1, 2, or 3, and submit a single assignment for the entire group on MarkUs.

1. Consider the following recursive linear search algorithm.

   \[
   \text{rec-lin-search}(A, i, x):
   \begin{align*}
   \text{if } & i == \text{len}(A): \\
   & \text{return False} \\
   \text{else:} \\
   & \quad \text{return } A[i] == x \text{ or rec-lin-search}(A, i + 1, x)
   \end{align*}
   \]

   (a) Define a recurrence \( T(n) \) for the worst-case run time of \( \text{rec-lin-search}(A, i, x) \), where \( n = \text{len}(A) - i \).

   (b) Use unwinding to find a closed form for \( T(n) \). Based on the closed form, state a conjecture for \( f(n) \) such that \( T(n) \in \Theta(f(n)) \).

   (c) Prove the bound on \( T(n) \) you gave in part (b) is correct.

2. Consider the problem of finding the maximum sum of consecutive values in a sequence of integers. For example, the maximum consecutive sum in \( 2, -5, 3, -1, -1, 4 \) is \( 3 + (-1) + (-1) + 4 = 5 \), and the maximum consecutive sum in \( 1, -2, 3 \) is 3. Let \( n \) be the number of integers in the sequence.

   (a) Give a brute force algorithm that solves the problem with worst-case running time \( O(n^2) \).

   (b) Informally state why your algorithm would have worst-case running time \( O(n^2) \).

   Consider a Divide & Conquer approach to this problem, based on the idea that, for any sequence, either the middle element is included in the maximum consecutive sum, or it is not.

   (c) Based on the idea above, consider how many subproblems the original problem would be split into, and approximately how large each subproblem would be. Then, consider a recurrence \( T(n) \) that represents the worst-case running time for an algorithm based on this idea. If the Master Theorem was applied to this algorithm, what would \( a \) and \( b \) be for \( T(n) \)?

   (d) Let \( f(n) \) be the cost of splitting and recombining from \( T(n) \), and suppose \( f(n) \in \Theta(g(n)) \). For this D&C algorithm to perform better than the brute force approach in part (a), what must \( g(n) \) be?

   (e) Give a D&C algorithm for this problem, with worst-case runtime \( T(n) \) that has \( a \), \( b \), and \( f(n) \) as described in parts (c) and (d).

   (f) Prove that \( f(n) \in \Theta(g(n)) \).

   (g) Use the Master Theorem determine \( h(n) \) such that \( T(n) \in \Theta(h(n)) \).
3. Recall that gcd(x, y) is the greatest common divisor of x and y, for all natural numbers x, y. Consider the following recursive algorithm to compute gcd(x, y).

REC-GCD(x, y):
    Precondition: x, y ∈ N.
    Postcondition: the algorithm returns gcd(x, y).
    if x = 0 or y = 0: return x + y
    if x = 1 or y = 1: return 1
    if x is even:
        if y is even: return 2 × REC-GCD(x/2, y/2)
        else: return REC-GCD(x/2, y)
    else:
        if y is even: return REC-GCD(x, y/2)
        else:
            if x < y: return REC-GCD((y - x)/2, x)
            else: return REC-GCD((x - y)/2, y)

(a) Give a detailed analysis of the algorithm's worst-case running time. (Start from a recurrence relation, show your work to obtain a closed form, and give a rigorous proof by induction that your closed form is correct).

Let n be the total number of bits to represent both x and y (i.e., n = n_1 + n_2 where it takes \( n_1 \) bits to represent x and \( n_2 \) bits to represent y). Also, assume that it takes linear time to perform basic arithmetic operations (like + and -).

(b) Write a detailed proof by induction that the algorithm is correct.

Hint: Take the time to write down a precise definition of “greatest common divisor” (gcd) and clearly state the properties of gcd you need to use in your proof.