1. (Lecture Questions)
   (a) Prove the Correctness for Iterative Merge, Recursive Merge, and for Merge Sort.
   (b) Find the recurrence relation for “Binary Strings Avoiding 00”.

2. 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) At a party with \( n \) people, everyone shakes hands once with everybody else. Let \( H(n) \) be the total number of handshakes among the \( n \) guests. Write a recurrence for \( H(n) \). Hint: Obviously, \( H(1) = 0 \) and \( H(2) = 1 \).
   (b) Let \( B(n) \) be the number of length \( n \) bit sequences that have no three consecutive 0's. Write a recurrence for \( B(n) \).
   (c) Say you are tiling a \( 2 \times n \) rectangle with L-shaped tiles of area 3 (trominoes).

   To tile the rectangle is to cover it with tiles so that no tiles overlap, no tiles are hanging off the edge of the rectangle, and every space on the rectangle is covered by some tile. Let \( T(n) \) denote the number of ways to tile the rectangle. Write a recurrence for \( T(n) \).
   (d) 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 \( T(n) \).

3. (a) Suppose a function \( g \) is defined by the following recursive formula, where \( n \) is a positive integer.

   \[
   g(n) = 3g(n - 1), \quad g(1) = 9
   \]

   Use the guess-and-check method to get a closed-form formula for \( g(n) \). That is, guess a formula for \( g(n) \) and use induction to prove that your guess is correct.
   (b) Suppose a function \( f \) is defined by the following recursive formula, where \( n \) is a positive integer.

   \[
   f(n) = f(n - 2) + 4, \quad f(1) = 1, \quad f(2) = 3
   \]

   Use the guess-and-check method to get a closed-form (i.e. not recursive) formula for \( f(n) \). That is, guess a formula for \( f(n) \) and use induction to prove that your guess is correct.
4. Suppose a function \( g \) is defined by the following recursive formulas, where \( n \) is a positive integer. Use the unraveling method to get a closed-form formula for \( g(n) \).

(a) \( g(n) = 3g(n-1), g(1) = 9 \)
(b) \( g(n) = g(n-1) + 2n + 1, g(1) = 3 \)
(c) \( g(n) = 3n \times g(n-1), g(1) = 1 \)
(d) \( g(n) = g(n-2) + 4, g(1) = 1, g(2) = 3 \)

5. The following algorithm (Rosen pg. 363) is a recursive version of linear search, which has access to a global list of distinct integers \( a_1, a_2, \ldots, a_n \).

\begin{verbatim}
procedure search(i, j, x: i, j, x integers, 1 \leq i \leq j \leq n)
1. if a_i = x then
2. return i
3. else if i = j then
4. return 0
5. else
6. return search(i + 1, j, x)
\end{verbatim}

(a) Prove that this algorithm correctly solves the searching problem when called with parameters \( i = 1 \) and \( j = n \). That is, prove that it returns the location of the target value \( x \) in the list, and returns 0 if the target is not present in the list. Hint: Use induction on \( k = j - i + 1 \).

(b) Let \( T(n) \) be the running time of this algorithm. Write a recurrence relation that \( T(n) \) satisfies.

(c) Solve the recurrence found in part (b) and write the solution in \( O \)-notation.

6. 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 \( racecar \).

\begin{verbatim}
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
\end{verbatim}

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) 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) Let \( C(n) \) be the number of times this algorithm compares two letters \( s_i \) and \( s_j \) for some \( i, j \). Write a recurrence relation that \( C(n) \) satisfies. Hint: \( C(0) = C(1) = 0 \).

(c) Solve the recurrence found in part (b) and write the solution in \( O \)-notation.

7. Say we are running an algorithm with running time \( \Theta(f(n)) \) for the function \( f : \mathbb{N} \rightarrow \mathbb{N} \). For each \( f(n) \) listed below, write down the multiplicative factor we suffer in running time when running the algorithm with input of length \( 5n \) instead of \( n \).

(a) \( f(n) = 5n \)
(b) \( f(n) = 4n^3 \)
(c) \( f(n) = 3^n \)
(d) \( f(n) = 5n^2 \)
(e) $f(n) = n!$
(f) $f(n) = n^n$

8. Prove the following $f(n) \in O(g(n))$ by providing witnesses, $k, C$ such that $f(n) \leq C \cdot g(n) \forall n \geq k$.

(a) $n^5 + 3n^2 + 13 \in O(n^5)$
(b) $n \log^3 n + 5n \log n \in O(n \log^3 n)$
(c) $4n \log n + n^2 \log(\log(n)) \in O(n^2 \log(\log(n)))$