1 Written Part (30 points)

1. R-4.12, p. 182
2. R-4.13, p. 182
3. R-4.15, p. 182
4. Consider \( f(n) = 3n^2 + 2n - 1 \), mathematically show that \( f(n) \) is \( O(n^2) \), \( \Omega(n^2) \), and \( \Theta(n^2) \).
5. For finding an item in a sorted array, consider “tertiary search,” which is similar to binary search. It compares array elements at two locations and eliminates 2/3 of the array. To analyze the number of comparisons, the recurrence equations are \( T(n) = 2 + T(n/3) \), \( T(2) = 2 \), and \( T(1) = 1 \), where \( n \) is the size of the array. Explain why the equations characterize “tertiary search” and solve for \( T(n) \).
6. To analyze the time complexity of Code Fragment 5.11 on p. 213, we would like to count the number of checks via “if \( S \) solves the puzzle”. Explain the number of checks in terms of \( m \) (number of symbols/variables) and \( n \) (number of digits).

2 Programming Part (70 points)

A palindrome is a string that reads the same forward and backward (excluding spaces and punctuations); for example, “wow” and “taco cat”. Finding multi-word palindromes can yield interesting phrases.

The goal of the assignment is to find multi-word palindromes. Words in a multi-word palindrome are unique. The two main components of the problem are:

1. Design a recursive algorithm that checks if a string is a palindrome or not.
2. Given a list of words in a file and \( \text{palLength} \) (number of words in a palindrome), design a recursive algorithm to find palindromes with \( \text{palLength} \) words.

Suggestion: Solve the single-word problem first (\( \text{palLength} = 1 \); one recursive algorithm), before solving the multi-word problem (\( \text{palLength} \geq 1 \); by adding a second recursive algorithm).

Input: Input is from the command-line arguments for HW2.java in this order:

1. filename of the list of words, one on each line
2. \( \text{palLength} \) (number of words in a palindrome), which is a positive integer

Output: The program prints palindromes in alphabetical/lexicographical order to the standard output (screen). Each palindrome is on a line.

Extra Credit (10 more points): Separate submission via HW2Extra.java. Solve the *entire* problem without recursion (or using a stack to simulate recursion).

3 Submission

Submit HW2.java that has the main method and other program files. Submissions for Individual and GroupHelp have the same guidelines as HW1.

Submit the written part in PDF format to the Submit Server. Hardcopy is also acceptable in the lab. GroupHelp submission is not applicable to the written part.

Note the late penalty on the syllabus if you submit after the due date and time as specified at the top.

For extra credit, submit HW2Extra.java that has the main method and other program files. GroupHelp submission is not applicable to extra credit. Late submission for extra credit is not accepted.