Mayank Vanjani
Homework 2
2/5/17

C-4.9
Runtime: O(n)
Without any recursive calls, the rest of the code runs at O(1) speed consisting of "if" statements and modifying the curr, minim, and maxim variables. Since each run of the function has only 1 recursive call and each recursive call moves the curr pointer 1 to the right, there are n recursive calls. N recursive calls with each call being O(1) results in an O(n) algorithm.

C-4.10
Runtime: O(logn)
Without any recursive calls, the code runs at O(1) speed because of the two "if" and "return" statements. There is only one recursive call per function and each recursive call lessens the number by an integer division of 2. This results in a runtime of O(logn) because the summation goes form level 0 to level logn while adding the number of calls times the runtime per call which is just 1. The runtime becomes O(log base 2 of n)

C-4.11
Runtime: O(2^n)
The function is O(1) per function call but there are 2 recursive calls per function which only make the size of the overall list decrease by 1. This results in a summation of 0 to n (because there are n levels when you keep decreasing the size by 1) of 2^level * 1 (2^l number of calls and O(1) per call). This ends up with a runtime of O(2^n).

C-4.15
Runtime: O(n^2)
Without any recursive calls, the function still implements list splicing (an O(n) algorithm) and list comprehension. The list comprehension loop is also an O(n) algorithm resulting in an O(n) runtime per function call without the recursion. There is only 1 recursive call per function and each recursive call shortens the list by 1. The runtime is then defined by the summation from 0 to n of 1 recursive call per function times the O(n) runtime per function. This results in an O(n^2) algorithm

C-4.17
Runtime: O(n)
The function has an O(1) runtime without the recursive call. When the recursion is called, the string being checked has its size reduced by 2 because you move the start and end counters inward 1 position each. This results in (n-1)/2 recursive calls with each recursive call taking O(1) time; this makes the total runtime O((n-1)/2) which simplifies to O(n/2) and further simplifies to O(n).

C-4.20
Runtime: O(n)

The function is O(1) without the recursive part. There is only 1 recursive call per function because the recursion is in an if/else statement. Each recursive call eliminates 1 element from the whole list and places it correctly in the answer list. This results in n recursive calls each with a runtime of O(1) resulting in a total runtime of O(n).