**CS 32 Homework 4**

**Problem 2**

The reason why the call to Sequence<Complex>::insert causes at least one compilation error is because that member function relies on the comparison operator > in order to determine where to insert the indicated value. Since the comparison operator > was not defined for the class Complex, the compiler is unable to perform the comparison and throws a compilation error. As a side note, the reason why the two-argument form of Sequence<Complex>::insert works is because it does not rely on any comparison operator since the user tells where to insert the indicated value.

**Problem 3b**

It is not possible to implement the one-parameter listAll as recursive because with only one parameter it is not possible to pass in a simplifying case to the recursive call each time. For example, if we assume the one parameter is const MenuItem\* m then we could visit each node but we would not be able to print out the corresponding path to each item. Thus with only one-parameter this problem is not solvable due to the inability to pass in an appropriate simplifying case each time.

**Problem 4a**

The time complexity is O(N^3). This is because we have a for loop nested inside a for loop nested inside a for loop, and each for loop will run N times therefore the time complexity is N (outer loop) times N (inner loop) times N (innermost loop) or N^3.

**Problem 4b**

The time complexity is O(N^3). This is because we still have a for loop nested inside a for loop nested inside a for loop, and even though each loop does not run N times each time, in the worst case they all will. Since time complexity is supposed to be based off the worst case scenario, the time complexity is N (outer loop) times N (inner loop) times N (innermost loop) or N^3.

**Problem 5a**

The time complexity is O(N). This is because the function has two independent for loops, which will visit every Linked List Node of seq1 and seq2. Since seq1 and seq2 both have N elements and each element in both sequences is visited once, the time complexity is technically O(2N) but we drop the coefficient. Additionally the function calls another function swap(), but the time complexity of that function is O(1) so it can be ignored. Thus when we account for all of this, we get a time complexity of O(N).

**Problem 5b**

The time complexity is O(N). This is because the function has two independent for loops, which will visit ever Linked List Node of seq1 and seq2 once. We then drop the coefficient to get our answer of O(N). Also again the swap() function runs in constant time so it can be ignored. In terms of performance, this algorithm is the same as the one in part a.