2. In order to successfully call the insert member function in Sequence class, the Complex class has to have an operator > in order to successfully compare two different Complex class objects. If there is no operator > in Complex class, there will be an error.

3b. It is impossible to solve this problem given the constraints in part a if we had only a one-parameter listAll function, and we have to implement this as a recursive function. The reason is that since it is a recusive function, we have to break this problem into two subproblems. Therefore we need to keep track of two thing. First is the current menu/submenu (we track this by having a menuItem pointer). At the same time, we need to edit the path of the menu. In order to do that we have to add the second parameter, a string, to denote the path of the menu item.

4a. The time complexity for this algorithm is O(N^3) because there is three levels of loops running from 1 to N. For each loop, we would have a time complexity of O(N). Since all of the loops are embedded, we would have a total time complexity of O(N \* N \* N) = O(N^3).

4b. The time complexity of this algorithm would still be O(N^3). In the worst case scenario, we see that I in the second loop(j-loop) would be N so that the time complexity for this loop is still O(N), like the one in part (a). Therefore, following the logic in part a, we would have time complexity of O(N \* N \* N) = O(N^3) for this algorithm.

5a. The time complexity for this interleave function is O(N^2). The reason is that, first of all, for loop (both two, because they are not embeded) would have O(N) in the worst case scenario, in which the two sequences have the same size, or there is an empty sequence. Moreover, we observe that inside the for loops, we call get() function, which has O(N) complexity. Combining the reasoning together we would have time complexity O(N\*N) = O(N^2) for this algorithm.

5b. The time complexity for this version of interleave function is O(N), better than the last version. The reason is that, for loops would have O(N) in the worst case scenario, just like last version. However, the insertBefore function called in the for loop only takes constant time complexity, O(1), since it just inserts the value before a designated node. Therefore, we have a total time complexity of O(N).