2. The one-argument form of Sequence<Complex>::insert needs to compare Complex objects to place the Complex object passed in before the first object greater than it. This causes a compilation error because it relies on the ‘>’ operand for comparison, but no comparison operands are defined for the Complex class.

3b. When listing out a path name for some menu item, it needs to be known where it came from, all the way to the original root menu. This is what the two-argument overload allows us to do. Without the second argument, the recursive function cannot know, for example, that “Window” came all the way from the “File” menu. It will only at most be able to print one menu back (“New/Window” instead of “File/New/Window”).

4a. The time complexity of this algorithm is O(N3). Starting from the body of the innermost for-loop, we have O(1) complexity. The innermost for-loop runs N times, so we have O(N) complexity. Going to the next for-loop, we have a similar case resulting in O(N2) complexity. Going to the outermost for-loop, we have O(N3) complexity. This does not change when taking into account the rest of the code, so the algorithm has time complexity O(N3).

4b. The time complexity of this algorithm is O(N3). Starting from the body of the innermost for-loop, we have O(1) complexity. Going to the innermost for-loop, we have O(N) complexity, same as above. The body of the next for-loop is gone through (0 + 1 + 2 + … + N-1) = N(N-1)/2 times. This results in a total complexity of O(N3) again.

5a. The time complexity of this function is O(N2). The body of the first for-loop has complexity of O(N) due to the get and insert functions using the O(N) complexity function nodeAtPos. Taking into account the for-loop, this becomes O(N2). It is the same for the second for-loop (though if everything has size N, this for-loop’s body is never entered). The swap function has complexity O(1). Adding everything together results in a time complexity of O(N2).

5b. The time complexity of this of this function is O(N). The first for-loop’s body has O(1) complexity as insertBefore has O(1) complexity. This for-loop’s body is performed some k number of times. The second for-loop’s body also has O(1) complexity for the same reason as before, and it is performed about N-k times. The swap function also has O(1) complexity. Adding everything together, we get O(N) complexity. This is better than the implementation in part a.