2.

The call to the one-argument form of Sequence<Coord>::insert causes build-time errors because the compiler tries to compare the values of Coordinates in the list using the operator>, but it doesn’t have a way to do that because there is no operator defined for Coordinates.

4.b)

You could not solve the problem because listAll needs some way to keep track of the full path name. By using a one-parameter listAll (without using an additional container such as a stack or using a global variable), we would have no way to pass the current path to new recursive calls.

5a)

The time complexity of this algorithm is O(N^3). The outermost for loop runs N times, the middle for loop runs about N times in the worst case, and the innermost for loop runs about N times in the worst case. The number of basic operations in each loop is trivial, since it does not depend on N and thus is a lower-order term. Therefore, the highest-order term will be O(N\*N\*N) = O(N^3).

5b)

The time complexity of this algorithm is also O(N^3). The outermost for loop runs N times, the middle for loop runs about N times in the worst case (even though the loop limit is such that the number of iterations is slightly less than it was in 5a, the worst-case highest-order term is still N), and the innermost for loop runs about N times in the worst case. The number of basic operations in each loop is trivial, since it does not depend on N and thus is a lower-order term. Therefore, the highest-order term will be O(N\*N\*N) = O(N^3).

6a)

The time complexity is O(N^2) in terms of the number of ItemType objects visited in the linked list nodes. The first for loop runs about N times and in each iteration, the highest-order time complexity is O(N) because when calling get() from a linked list, in the worst case (where the node you want to access is at the very end of the list), you have to visit all N of the nodes in the list. The second for loop runs about N times and in each iteration, the highest-order time complexity is O(N) because when calling get() from a linked list, in the worst case (where the node you want to access is at the very end of the list), you have to visit all N of the nodes in the list. Since each for loop’s total time complexity is O(N^2), and that is the highest-order term of the entire function, the time complexity is O(N^2).

6b)

The time complexity is O(N) in terms of the number of ItemType objects visited in the execution of the function. The first for loop runs about N times, since it runs until one or both of the linked lists “run out of elements” (indicated by their node pointer pointing back to their head pointer), and each iteration of the loop, we do two insertions which are only O(1). The second for loop, which traverses the excess nodes of the longer linked list and inserts them into the resulting sequence, does not run a significant number of times since both sequences are about N items long. Therefore, the overall, highest-degree time complexity of the function is O(N).