Problem2:

Because there is not < operator defined for class Coord.

When you call the insert function on Coord, the insert function calls FindAtLeast function, which is going to use the < operator on Coord, which is undefined behavior.

Even though there is an < operator, when it keeps through and comes to the == operator, we will have another undefined operator for Coord.

Problem4b:

Because we can only go downwards in such a tree. So if we do not pass the previous labels (string path) as parameter, there is no way that we can go back and output them after we are advance to the next depth.

Problem5a:

O(N^3), because we run the codes in under the brackets of the most inner “for loop” N(for i)\*N(for j)\*N(for k) times.

Problem5b:

O(N^3), because after we omit the coefficient, the difference caused by new “j” for loop is ignored. Time complexity in the new code is N\*N/2(for i and j)\*N(for k), which then become the same as O(N^3).

Problem6a:

Worst case time complexity: O(N^2)

Because the only place where linked list nodes are accessed is the for loop

The get function is called N times in the for loop, with parameter from 1 to N, so it’s time complexity is (N^2+N)/2; the insert function is called N times, with parameter in the worst case causing the nodes to be accessed for N to 2N times, so it’s time complexity is (3N^2/2). As a result, the time complexity is O(N^2).

Problem6b:

Time complexity: O(N log N)

When we copy all items into v, we visit nodes in linked list for 2N total times.

When we sort v using an O(N log N) algorithm, time complexity is N log N.

When we delete result nodes, time complexity is N.

When we copy the unique items from v into result, time complexity is N.

As a result, time complexity is O(N log N).

Problem6c:

Time complexity: O(N)

In the while loop, the time complexity is O(N).

In the for loop, the time complexity is also O(N).

As a result, the time complexity is O(N).