Homework 4

2) The call to the one-argument form of Sequence<Coord>::insert results in a compilation error because the operator > is not defined for the Coord class, so the compiler has no way of comparing two Coord class objects to see which one is larger. The call to the one-argument form of Sequence<int>::insert was not an issue because objects of type int have a predefined > operator that the compiler can use in the insert function, and the call to the two-argument form of Sequence<Coord>::insert was not an issue because it does not compare multiple Coord objects.

4B) Given the constraints in part A, the one-parameter overload of listAll could not be implemented as a recursive solution to the stated problem because it would be impossible for any given node to know about anything above it in the tree, so a leaf would have no way of knowing and printing the path taken to get to it. Therefore, the path would have to printed at each level of nodes, which makes it impossible to achieve the desired output because then the path would be printed in the opposite order (e.g., cs.ucla.edu would incorrectly be printed as edu.ucla.cs). Thus, it is necessary to have at least 2 parameters so that the path taken to a leaf can be updated at each level of nodes, and it can be printed in the proper order once a leaf is reached.

5A) The time complexity is O(N^3), or “O of N to the third power”. This is because the algorithm uses three nested for loops, each of which runs a maximum of N iterations (note in particular that the maximum number of iterations of the middle loop for (int j = 0…) is N-1; therefore, we consider this middle loop to run for a fixed number of iterations equal to the maximum possible iteration count N-1). Therefore, the outer loop (i.e. for (int i = 0…) ) runs for a total of N times, the middle loop (i.e. for (int j = 0…) ) runs for a total of N\*(N-1) times, and the inner loop (i.e. for (int k = 0…) ) runs for a total of N\*(N-1)\*N times. So the overall time complexity, discarding the lower-order N and N^2 terms, is O(N^3).

5B) The time complexity is still O(N^3) for the same reasons as those stated in part 5A. Although the algorithm was altered to take advantage of symmetry, the original portion of the algorithm that was replaced ran cN^2 times (where c > 0), and the new portion of the updated algorithm runs kN^2 times (where 0 < k < c). The difference is thus negligible because both portions run on the order of N^2 times, where N^2 is a lower-order term in the algorithm’s overall time complexity. Thus, the time complexity is still O(N^3).

6A) The time complexity is O(N^2). This is because, based on the assumptions given in the problem statement, the first for loop (i.e. for (int k = 0…) ) runs a total of N times, since N is the size of both seq1 and seq2. Within that loop, each of the Sequence class member functions (i.e. get and insert) have a time complexity of O(N). Therefore, the first loop contributes a time complexity of O(N^2). The swap function, constructor, and assignment statements are O(1), which is a lower-order term that can be disregarded. Lastly, the second for loop (i.e. for (int k = nmin…) ) does not iterate, based on the problem’s assumption that seq1 and seq2 have the same size N. Even if it did run N times, it calls the same Sequence member functions as the first for loop, so it would contribute another N^2 term that would not change the time complexity. For these reasons, the overall time complexity is O(N^2).

6B) The time complexity is O(N). This is because the first for loop (i.e. for ( ; p1 != seq1…) ) runs N times because it iterates through all of the elements of seq1 and seq2. The body of the loop calls the member function insertBefore, which is O(1), so the time complexity of the loop is O(N). Outside of the loops, the constructor, assignment statements, and swap function are all O(1), so they are lower-order terms that can be discarded. The second for loop (i.e. for ( ; p != pend…) ) does not run because seq1 and seq2 are both assumed to have N elements. Even if it did run N times, the body of the loop only calls the insertBefore function, which is O(1), so the loop would be O(N) and thus would not change the time complexity of the algorithm. Because of these considerations, the overall time complexity of the function is O(N).

This implementation is better than the implementation in part 6A because the time complexity is lower (O(N) in part B vs O(N^2) in part A), so it is more adept at processing large sequences.