Homework 4

2. Attempting to call Map<Coord, int>::insert leads to a compilation error because the Coord class doesn't define (have an overloaded) != comparison operator, but the insert method is using the !=operator to compare KeyType values. When using a Map, and the left-hand side, is a user-defined class, there needs to be the inclusion of a definition of a comparison operator (in this case the !=), if it were to be used.

3.b. You wouldn't be able to solve this problem because there is only one parameter, and without the string parameter, there would be no way to keep track of the complete path. Each recursive call would only allow you to know the current path, but there would be no way to cout the entire path string.

4.a. The time complexity of this algorithm is O(N^3). The reason is because there are two nested for-loops, and each run a set of basic operations for their bodies. Every iteration of the body of the innermost for-loop does a bunch of operations (comparison, array element access, iteration etc.) that takes constant time (lets call it c). The innermost loop (the k-loop) is proportional to N and will run N times (lets call it cN). Every iteration of the middle loop (the j-loop) also takes the number of steps that are proportional to N plus some constants (lets call it cN). So the j-loop is being executed N times (therefore, it is running cN, N times) or rather, cN^2. Similarly, every iteration of the outermost loop (the i-loop) takes the number of steps proportional to N plus some constants (so cN^2). So, in conclusion the i-loop is being executed N times(therefore, it is running cN^2, N times) or rather cN^3. We only care about the highest order, and can drop the constants, so it is simply order of N^3 or O(N^3).

4.b. The time complexity of this algorithm is still O(N^3). This is because it uses similar logic to the previous example, the only thing that was changed was the middle loop (the j-loop). The middle loop will run 0 iterations if i = 0, 1 iteration if i = 1, … and if i = n – 1, it will run n – 1 iterations. Therefore, the middle loop is equal to n(n – 1)/2 or N^2 (by removing lower order terms and constants). However, the only loop that is affected by the change is the j-loop, but only the coefficient is changed, the order still comes out to cN^2. But, because we ignore the constants for large N, the time complexity comes out to the same order of N. Therefore the time complexity still comes out to O(N^3).

5. The outer for-loop will run a total of N times. Every iteration of the loop results in the 3-parameter get function being called. This get function has a time complexity of N, after dropping the constants. Every iteration will then call the 2-parameter get function, and either call the insert function or the erase function. These three functions all have a time complexity of N because they call the find function, which has a for-loop, and runs N times due to it running a linear search. Thus every iteration of the for-loop has a time complexity of cN (this can be done by adding up the total time complexities of each iteration, along with the basic operations ( i.e. initialization, iterations, comparisons etc.)). Swap will also be called, but this runs in constant time, because its execution is independent of the list size. The total complexity comes out to N^2, because the for-loop is executed N times (running cN, N times), or rather N^2. Therefore, the time complexity is O(N^2).