# Homework 4 Doc

#### 2.

The call to Set<Coord>::insert causes two compilation errors due to the fact that the insert function and the findFirstAtMost function (called inside the insert function) have operators for comparison like > and == which aren’t implemented as custom operators inside the Coord class. Therefore, the compiler cannot interpret what to compare when calling the Coord class, and instead just compares the Object as a whole which throws a compilation error.

#### 4b.

If the problem given only had a one-paramater listAll, it would be impossible to solve this problem as the function would have no way of storing the previous path of the menu object. This means that the program wouldn’t be able to store and then print the values before the final item of the menu.

#### 5a.

This algorithm has a nested for loop inside of a nested for loop, and each loop compares every term from 0 to N, hence the time complexity of this algorithm is N \* N \* N = N^3 🡺O(N^3)

#### 5b.

This is yet again another nested for loop inside of a nested for loop, but this time, i will iterate till N, j won’t iterate when i = 0 and j will reach a value of N-2, and k won’t iterate at the points when j doesn’t iterate but it will reach a value until N. Hence, we can calculate that the time complexity of this algorithm is N \* (N-3) \* (N) = N^3 – 3N^2 🡺 O(N^3)

#### 6a.

Unite iterates through the size of a linked list sp so its time complexity is N, the get function doesn’t iterate over the linked list nor uses any algorithm so its time complexity is 1, and then the insertion function calls the findFirstAtMost function which also loops through the linked list so there is a nested loop and therefore its time complexity is also N. Hence, the worst case time complexity is N \* N \* 1 🡺 O(N^2)

#### 6b.

This unite function has two loops which are both of time complexity N, with each push\_back being having time complexity 1 each. Hence, we are left with 2N + 2. The sort algorithm is of time complexity NLog(N) as given by the comments. The while loop calls the doErase function N times which means that its time complexity must be N \* time complexity of doErase (which is 1 because no loops or algorithm is used), therefore its N. This is the same case for the for loop, but since the loop is iterating over the total of the two sets 1 and 2, its time complexity is 2N \* time complexity of insertBefore (which is 1 because no loops or algorithms is used). Hence, in total, the time complexity for this algorithm is Nlog(N) + 2N + N + 2N + 2 🡺O(Nlog(N))

#### 6c.

This unite function doesn’t use any algorithms beside the two loops which both iterate N times for each loop and the insertBefore call in the second for loop which has a time complexity of 1. Hence, we have N + N + 1 = 2N 🡺 O(N)

#### 