**Aditya Pimplaskar**

**UID: 804900470**

**Mar 1, 2019**

**HW 4**

2. Set<Coord>::insert causes an error due to the absence of an operator< definition (or any other comparison operator) for the Set container to use to order Coord objects (and ensures their uniqueness) when it inserts. For primitive types or types like string whose library has a defined operator<, this will not produce an error as there are defined criteria for sorting/ordering these elements; since the Coord class lacks such criteria, we will see an error.

3b. A one parameter definition of this function will fail to take into account the path leading up to a particular set of menuItems. In turn, we won’t necessarily be able to get a full path given we won’t know the steps taken leading up to a particular item being printed out.

4a. This should be **O(n^3).**

We have three nested for loops that go from 0 to N.

4b. This should also be **O(n^3).**

Even though the middle loop goes from 0 to i, I’s maximum value is N. In turn, in the maximum case, the middle loop still runs N times; this combined with the N times each of the other two loops runs yields O(n^3).

5a. This is **O(n^2).**

The get function should be O(n); a worst case of n/2 steps Is still order n. The insert function should also be O(n). Since we loop over N items, and the body of the loop runs 2n times, we get order n^2.

5b. This is **O(n log n).**   
Pushing p1 and p2 is 2n steps which is order of n. We are given that sorting takes nlogn. The insertBefore function is O(1). Looping over the elements and deleting is also O(n). We can ignore the n and 1 terms as the n log n term is the highest order.

5c. This is **O(n).**

The conditionals at the beginning that compare pointers are O(1). In the best case, we will satisfy the first set of conditionals and not have to run any loops, making the runtime O(1). In the worst case, we traverse down n elements in s1 and loop over n elements in s2. The loop’s body has an insertBefore function, which is O(1) so it does not contribute to the order of the time complexity. This should give us 2n, which is of order n.