**2**

Because Set<Coord>::insert internally comparing the Items and inserting the new Item

at appropriate position. C++ already knows how to compare built-in types like int and double,

but it doesn't know how to compare a user defined class Coord. In order to fix the error,

we must tell C++ how to compare Coord object by defining operator < and operator == functions for Coord class.

**3b**

It would be possible if each Menu only have at most one submenu. But since it can have more than one submenus, we need one more parameter to keep track of directories.

**4a**

Since we only care about the high order term, and we ignore the scaler, we can just consider the worst case. The i will loop at most N times. Similarly, j and k will loop at most N times. Therefore, N \* N \* N = O(N^3).

**4b**

Here again, we only consider the worse case. The i will loop at most N times. The j will

loop at most N times. (Although technically j will loop 1 + 2 + ... + N times, but we will

only think of the worst case.) The k will loop at most N times. Therefore,

N \* N \* N = O(N^3)

**5a**

Since we only care about worst case, we can only focus on the time complexity of the for loop. The for loop will run N times. sp->get(k, v) is doing linear search, so it takes O(N).

(Technically it is O(N/2) because it will find the element before reaching the middle.)

result.insert(v) first call findFirstAtLeast() and insert v in front of it. findFirstAtLeast()

simply loops through the entire set from beginning to the end. The worst case for findFistAtLeast() is N. The insertion itself just takes constant time. Therefore, result.insert(v) takes O(N). Sp, the total time it takes is N \* (N + N). Thus, the time complexity unite() is O(N^2).

**5b**

To copy all items into v from set1, it takes O(N) to loop through all the elements in set1.

Similarly, it takes O(N) to copy all items into v. sort() takes O(Nlog(N)). Deleting result nodes takes O(N) to loop through all the elements in result. Deleting takes constant time. Copying unique items from v into result takes O(N). (Technically it's O(2N)). insertBefore() takes constant time. Therefore, it takes N + N + Nlog(N) + N + N = O(Nlog(N)).

**5c**

While loop and for loop will loop through all the elements in set1 and set2 once. It takes

N + N = O(N).