**Question 2**

Since the template of function Sequence<ItemType>::insert calls operator >. In the function implementation, it assumes that the line value > p->m\_valuehas a valid meaning. But the operator > does not work when as both operands are Coord and there is no available definition that can overload operator > for Coord.

**Question 4b**

Since when implementing recursive function (without global variable and static member), the current file will have no way to know what its parent directories are if only pass in the parameter const File\* f (it is impossible to keep track the full path from the root to the current file). However, if we pass in two parameter, with a string containing its parent director. Then, recursive function would have no trouble telling its full path.

**Question 5a**

Consider the inner most loop, each loop contains several operations whose time complexity is O(1), since the loop operators N times, the overall time complexity is O(N). (total number of operation is kN, were k is some constant).

Consider the second inner loop. Each loop contains some operation that happens constant number of times, but each loop also contains an inner loop that happens kN times. So the overall time complexity is O(N2). (total number of operation is N\*(m + kN) = kN2+mN times, where m is a constant).

The out most loop contains two inner loops and some comparison & incrementing operations for each loop. So total number of operation is N\*(kN2+mN+n) = kN3+mN2+nN, where n is some constant. So, the overall time complexity is O(N3). Adding some operations outside the loop which happens some constant number of times, which does not change the overall time complexity, the time complexity of the whole algorithm is **O(N3)**.

**Question 5b**

The time complexity is still **O(N3)**.

Consider the inner most loop, each loop contains several operations whose time complexity is O(1), since the loop still operators N times, the overall time complexity is O(N). (total number of operation is kN, were k is some constant).

Consider the second inner loop. Each loop contains some operation that happens constant number of times, but each loop also contains an inner loop that happens kN times. Since the loop happens **i** times, total number of operation is i\*(m + kN) times, where m is a constant).

The out most loop contains two inner loops and some comparison & incrementing operations for each loop. So total number of operation is = , So, the overall time complexity is O(N3). Adding some operations outside the loop which happens some constant number of times, which does not change the overall time complexity, the time complexity of the whole algorithm is **O(N3)**.

**Question 6a**

In the first loop, each get() function calls nodeAtPos() function, which requires steps of operation when and require steps when . Each insert() function will also call nodeAtPos() function and an insertBefore() function that operates a constant number of times. Since the loop operates N times, the number of operations in this loop is

, which gives us a time complexity of O(N2).

The assignment operator calls the copy constructor, whose time complexity is O(N), and a swap function with O(1). So, the overall complexity is O(N).

The second loop in the worst case will have time complexity of O(N2), but in this case, it will operate 0 times.

Finally, the swap function operates with time complexity O(1).

Hence, combining all of these and adding some other operations in the function body like comparison, the overall complexity of this function will be **O(N2)**.

**Question 6b**

The first loop contains two insertBefore function and several constant amounts of operations each cycle. So, the time complexity is O(N). The second for loop contains one insertBefore function each cycle (and some incrementation and comparison). Therefore, its time complexity in the worst case is O(N) (but in this particular case, the loop runs 0 times). The swap function has time complexity O(1) and the function contains several other operations which happens constant number of times. Hence the overall time complexity is **O(N)**, better than the implementation in part a.