Joel George

CS 32

Homework 4

2. The less than operator has not been defined to determine which Complex class is greater than the other, and so the single argument insert function does not know where to actually insert the Complex object. The reason the other insert function works is because it assigns a position to insert the Complex object; there is no comparison involved, it is a simple positional insertion.

3b. Since this is a void function, there would be no way for the function to represent the path it took before this iteration of the function. It would only print out the current name it was at, because there was nothing to pass on information from previous recursions.

4a. N3, because the maximum number of times the outer loop can run is N times, while the maximum number of loops the second loop runs is N\*N, thus making the maximum number of loops on the N\*N\*N, or N3. Thus, the number of critical operations performed in the inner loop is given by *a*N3, where *a* is some constant. Since this is just Big ‘O’, we don’t need the constant, so the time complexity is simply N3.

b. N3, because the worst case of this function still makes the outer loop run N times, while the inner loop runs up to i2, which in the worst case is at N2, and this further makes the inner loop run N3 times. The average runtime of this function is probably much better, but the Big ‘O’ of the function does not change, because the worst case looks identical to the previous function’s regular case.

5a. N2, because the first loop, given that the size of both sequences is simply N, will run a number of N times. Furthermore, the get and insert functions calls nodeAtPos() each time they are called, and nodeAtPos() has a Big ‘O’ of N/2. This means that the get() and insert() functions adds N2/2 every time they are called. This is the biggest contribution to the time complexity of the function. Since the second loop runs 0 times (as k in this case is set to N, and can only run as long as it is less than N), there is no contribution from that loop. All contributions to runtime complexity from the other operations in that loop are negligible.

b. N, because both loops run a total of N times, and the insertBefore(), which is the only function called in those loops, has a constant contribution of 1 to the time complexity.