2.

The class Bottle doesn’t have an equals or not equals operator defined by default, so the compiler doesn’t know how to check whether two Bottles are equal or not equal. When the insert function is called, it calls the find function, which attempts to check whether two Bottles are not equal, thus causing an error. If an equals and not equals operators are defined, which perhaps check the contents of two Bottles for equality, then the function works correctly.

3b.

Since listAll is a void function, we need to have a variable to keep track of the path to the subdirectories and files. The second parameter string path tracks how far we have traveled into the directory and thus allows listAll to print the correct pathname from the beginning.

4a.

The time complexity of this algorithm is O(N3). The algorithm has 3 embedded for loops that run for N times each. All the other operations are constant or run for N times so they are dropped in the big O calculation.

4b.

The time complexity of this algorithm is still O(N3). The key difference in this algorithm is that the second for loop runs for as many times as the variable of the outmost for loop, which goes from 0 to N-1. Thus the first two loops run for about N2/2 - N/2 times, and the 3 loops altogether run for about (N2/2 - N/2)N times so the algorithm is still O(N3).

5a.

The worst case time complexity occurs when the address of result is not the same as the address of set1 or set2, so set1 is assigned to result, which requires the visiting of every node of set1 to insert its value into result and performs N operations. The for loop then runs for N times, each time getting the value from a node in the set pointed to by sp, which can require traversing up to half of the nodes in the set and N/2 operations. Discarding the lower orders, the time complexity is O(N2).

5b.

Copying all the items from set1 and set2 into the vector requires traversing all the nodes in the sets, and since inserting into the end of a vector is constant time, this requires a total of 2N operations. Sorting the vector and deleting the result nodes, which requires traversing all the nodes in result, adds another 2Nlog2N + N operations. The last for loop runs for 2N times, in the worst case accessing each element of the vector twice and inserting every element in the vector into result, all constant time operations. Dropping lower orders and coefficients gives a time complexity of O(NlogN).