2. Because the operator “!=” is not defined for the URL class. So when the find function is called in the insert function, compiler is unable to traverse through the linked list as the “!=” operator is used in the for loop.

3. b.We need the first parameter because it is used in both the basic case and simplifying step and only with it we can access the information in the menuItems. We need the second parameter because it contains the name of menuItems in hierarchical order and is where we add the slash. Since we need both parameters in recursion, we cannot do it with only one parameter.

4.

a.O(N3), because the outer loop runs for 1 time; the second loop runs for roughly N times for i is incremented from 0 to N-1; the inner loop runs N\*N times for j is incremented from 0 to N-1; operations in the inner loop runs roughly N\*N\*N times for k is incremented from 0 to N-1. So the big O is O(N3)

b. O(N3), because the inner loop runs at most for roughly N times as the largest possible i is N-1. Big O considers the worst situation. So Big O is still O(N3).

5.

a.O(N2), the operations in the loop runs for N times. The time complexity of the function get is O(N), because in the worst case it traverses through N/2 Nodes. The insert function has time complexity O(N) because it calls the find function which loops trough every node in the linked list. The swap function has time complexity of O(1) because it simply swap the head and tail pointers. So the total complexity is O(N2).

b. O(NlogN). Copy two multisets into one costs 2N. Sorting this multiset takes 2Nlog(2N). The for loop takes 2N steps, and the final do while loop takes at most 2N steps. So the algorithm is 2N+log(2N)+2N+2N. The Big O is O(Nlog(N)), which is better than the previous one.