2. The insert function makes a call to the find function passing it the URL.

The problem is that the find function attempts to compare two URLs with the != operator which has not been overloaded for class URL.

3b. You could not solve this problem because you would not be able to keep track of the path. For example if you made a local variable it would reset every recursive call so the function would not be able to cout the previous path.

4a. This is order N^3 because it does linear searches through the array 3 times. In other words each loop runs through N elements of the array and because these loops are nested inside one another you get order N^3.

4b. explanation in steps:

i) the inner most loop runs N times

ii) the outermost loop runs N times as well

iii) the middle loop will run zero iterations when i = 0, 1 iteration when i =1, 2 iterations when i =2,….when q = n-1 the middle loop runs n-1 iterations. This is equal to n(n-1)/2 which is order N^2.

iiii) Now taking the inner most loop of order N into account, we see that the entire algorithm is order N^3.

5a. The outer loop does a linear search of order N. Inside the loop, “get” is called and although it splits the list once, making its time complexity N/2, when you drop the constant of proportionality it is order N as well. “contains” and “insert” are both order N because they call the “find” function which does a linear search. The loop in “find” runs N times independently of the loop that encapsulates the function calls to “get”, “contains”, and “insert.” Swap is constant time because it just exchanges two pointers which is a functionality that is independent of the size of the list. Therefore the total complexity is N^2. (You essentially add the time complexities on the inside and then multiply that by the time complexity of the whole loop).

5b. This code is of order NlogN. This stems from the fact that the sorting algorithm in this code is of order NlogN. Since everything else in the function added and either of constant time or of order N, this can be ignored when figuring out the average time complexity. This code is ultimately better than the implementation in part A because it is smaller than order N^2.