Ryan Hartz – Assignment 1

Question 2:

I started working on code for this one, the fragments of which can be found in my java file, but I'm answering in text instead. I began by converting both strings to arrays to make them easier to iterate through. Then, a for loop to iterate through every element of the first array, and a nested for loop inside to iterate through the second array. As soon as the elements match, both arrays would be iterated through to find out how long they keep matching, subsequently, and the matching elements are added to a storage String. If the program finds another substring, the length would be compared with the existing stored String, and the longer would be kept.

Question 6:

1. O notation is O(n), because the most time consuming operation is my for loop, which costs O(n), and hence that becomes the time cost for the entire method. Omega notation is also Omega(n), because the for loop will run n times no matter what.
2. My answer for 2 given above has a for loop inside a for loop, so complexity of O(n squared). These both iterate through the entire array, so it has to be every element every time, so the omega notation is the same, omega(n squared).
3. Most intensive operation is my for loop, so O(n) complexity. This loop is also guaranteed to run for all of n, so Omega notation is Omega(n).
4. Most intensive is a for loop, so O(n). This one is not guaranteed to keep running, however. The for loop could potentially only run once, and the same can be said for the while loop earlier on, so the omega complexity of the function is constant.
5. The most intensive operation here is a while loop within a for loop, giving an O complexity of n squared. The while loop can become constant in the best case, while the for loop cannot, so the omega complexity is just n.