1. A few notable obstacles I overcame included figuring out the best way to loop through each element of arrays. It took me a while to figure out that for loops were the best way to check each element of the array. I also had trouble with value swapping, and did not realize that I needed a third variable in order to swap the values of two variables. Lastly, the split function was extremely challenging. I did not know how to approach the problem, or how to start to tackle the challenge. I fixed the problems by learning how to debug, and to cout statements within my loops in order to track the value of variables. This allowed me to pinpoint the problems of my code and to actually see what each iteration of my code actually achieved.

2.

string a[7] = { “Alana”, “Jerry”, “Ben”, “Jonathan”, “Zack”, “Hillary”, “Maggie” };

assert(appendToAll(a, 7, “!!”) == 7 && a[5] == “Hillary!!”);

Reason: checks if it returns the correct number, and tests to see if Hillary becomes Hillary!!

assert(lookup(a, 7, “Zack!!”) == 4);

Reason:Checks to see if program matches target with element in array

assert(lookup(a, 7, “Shirley”) == -1);

Reason: Because Shirley isn’t found in the array, checks if it returns -1

assert(positionOfMax(a, 7) == 4);

Reason: checks to see if it returns position of name that appears last in the alphabet

assert(rotateLeft(a, 7, 2) == 2 && a[3] == “Zack!!”);

Reason: checks if returns pos parameter and checks for new position of elements

string b[5] = { “Alpha”, “Beta”, “Beta”, “Gamma”, “Delta” };

string c[6] = { “Alpha”, “Beta”, “Beta”, “Delta”, “Epsilon”, “Zeta” };

string d[3] = { “Delta”, “Epsilon”, “Zeta” };

string e[2] = { “Zeta”, “Eta” };

assert(countRuns(b, 5) == 4);

Reason: makes sure that Beta Beta is only counted as one run, checks if 4 is returned

assert(differ(b, 5, c, 6) == 3);

Reason: checks to see if it successfully catches that at position 3, the arrays differ

assert(differ(b, 2, c, 3) == 2);

Reason: Checks that if arrays are same up to length of smaller list, that it returns the smaller number

assert(subsequence(c, 6, d, 3) == 3);

Reason: checks if it returns position of beginning of the subsequence

assert(subsequence(c, 6, e, 2) == -1);

Reason: checks to see if it returns -1 after not finding a subsequence

assert(lookupAny(c, 6, e, 2) == 5);

Reason: checks to see if it finds “Zeta” in string c

assert(lookupAny(b, 5, e, 2) == -1);

Reason: checks to see if it returns -1 if no element of e is found in b

assert(flip(b, 5) == 5 && b[0] == “Delta” && b[3] == “Beta”);

Reason: Checks for return value and checks if array successfully flipped values

assert(split(a, 7, “Jack”) == 3);

Reason: checks to see that the position of the first string that comes after Jack is returned

assert(split(a, 7, “Zeck”) == 7);

Reason: checks to see if it returns n if no element is greater than or equal to “Zeck”.