**Project 4 Report**

**Notable Obstacles**

1. A method to make the separate function work without creating an extra array wasn’t immediately obvious to me.
2. Making sure to never go outside the bounds of an array was fairly important to maintain. I had to make sure that I addressed the possibilities for unusual inputs that may have caused my code to look beyond the array's boundaries. Certain zyBook excerpts helped me tweak my program and overcome this obstacle.
3. Designing test data to ensure functions worked properly was not easy on this project. It was difficult to build an all-encompassing set of test data.

**Test Cases**

**Using the following arrays for all the other functions**

string c[4] = { "ma", "can", "tu", "do" };

string hello[5] = { "h", "e", "l", "l", "o" };

string stuff[6] = { "mahi", "bon", "cous", "", "tar", "mur" };

string fruits[3] = { "apple", "banana", "orange" };

string names[8] = {"jeff", "allan", "kelsey", "kari", "elon", "buzz", "lewis", "sundar"};

string namesSubset[5] = {"kelsey", "kari", "elon", "Buzz", "Jeff"};

string names2[]={"jeff", "allan", "kelsey", "buzz"};

string num[10] = {"1", "2", "3", "4", "5", "6", "7", "8", "9", "10"};

string numberList[15]={"5", "3", "8", "2", "8", "3", "3", "6", "1", "1", "7", "8", "4", "7", "2"};

string e[10] = { "today", "today", "tomorrow", "", "", "", " ", "cs", "31", "31" };

**int reduplicate(string a[], int n);**

(c, 4) && c[0] && c[3] - expected: return 4, "mama", and "dodo", checking after duplication

(hello, 5) && hello[0] - expected: return 5 and "hh", checking equality of elements

(stuff, 6) - expected: return 6, checking after duplication

(fruits, 3) - expected: return 3, checking after duplication

**int locate(const string a[], int n, string target);**

(names, -10, "jeff") - expected: return -1, because the size is negative

(names, 0, "jeff") - expected: return -1, because "jeff" could not be found in 0 elements

(names, 8, "Jeff") - expected: return -1, because Jeff is capitalized

(names, 8, "0") - expected: return -1, because "0" is not in the array

(names, 8, "lewis") - expected: return 6, because "lewis" is in the sixth position

(names, 4, "lewis") - expected: return -1, because "schottie" is not in the first four positions

**int locationOfMax(const string a[], int n);**

(names, -10) - expected: return -1, because the size is negative

(names, 0) - expected: return -1, because the array has no elements

(names, 8) - expected: return 2, because "kelsey" is the "largest" string

(names, 2) - expected: return 1, because "allan" is the "largest" string out of the first two

(numberList, 15) - expected: return 2, because the both the 2nd, and the 11th position hold the biggest string, but 2 is earlier

**int circleLeft(string a[], int n, int pos);**

(num, 1, 0) - expected: return 0, because it is the only element, and array is unchanged

(num, -10, 1) - expected: return -1, because the size is negative

(num, 10, -1) - expected: return -1, because the position is negative

(num, 10, 20) - expected: return -1, because the position is bigger than the size

(num, 0, 0) - expected: return -1, because 0 elements means that no elements can be accessed to be "rotated"

(num, 6, 3) - expected: return 3, and elements 2 through 5 should be rotated left

(num, 10, 4) - expected: return 4, and elements 5 through 9 should be rotated left

**int enumerateRuns(const string a[], int n);**

(e, -2) - expected: n is negative

(e, 0) - expected: n is 0, should return 0

(e, 10) - expected: test for correctness

(e, 8) - expected: only the first 8 elements should be checked

(e, 1) - expected: only the first element should be checked

**int flip(string a[], int n);**

(num, -10) - expected: returns -1, because the size is negative

(num, 1) - expected: returns 1, and array is unchanged

(num, 0) - expected: returns 0, and array looks the same

(num, 5) - expected: return 5, and the first five elements are flipped

(num, 10) - expected: return 10, and the entire array is flipped

**int locateDifference(const string a1[], int n1, const string a2[], int n2);**

(names, -10, names, 8) - expected: returns -1, because size is negative

(names, 8, names, -10) - expected: returns -1, because size is negative

(names, 0, names, 8) - expected: returns 0, because first array differs/ends at position 0

(names, 8, names, 0) - expected: returns 0, because second array differs/ends at position 0

(names, 5, names, 8) - expected: returns 5, because the first array ends at element 5

(names, 8, names, 5) - expected: returns 5, because the second array ends at element 5

(names, 8, names, 5) - expected: returns 8, because the arrays are the same

(names, 8, num, 15) - expected: returns 0, because the second array differs from the first by the first element

(names, 8, names2, 4) - expected: returns 3, because the array differs at position 3

**int subsequence(const string a1[], int n1, const string a2[], int n2);**

(names, -10, names, 8) - expected: returns -1, because the size is negative

(names, 8, names, -10) - expected: returns -1, because the size is negative

(names, 5, names, 8) - expected: returns -1, because the second array is larger than the first

(names, 8, names, 5) - expected: returns 0, because the subsequence would start at the beginning

(names, 0, names, 8) - expected: returns -1, because the subsequence cannot be found in an array that is smaller

(names, 8, names, 0) - expected: returns 0, because the empty array is a subsequence of every array

(names, 8, names, 8) - expected: returns 0, because the subsequence would start at the beginning

(names, 8, num, 4) - expected: returns -1, because there is no match anywhere

(names, 8, namesSubset, 5) - expected: returns -1, because it differs at position 3

(names, 8, namesSubset, 3) - expected: returns 2, because the subsequence starts at 2

**int locateAny(const string a1[], int n1, const string a2[], int n2);(names,-10,names,8) - expected: returns -1, because the size is negative**

(names, 8, names, -10) - expected: returns -1, because the size is negative

(names, 8, names, 0) - expected: returns -1, because there is nothing to match

(names, 0, names, 8) - expected: returns -1, because there is nothing to match

(names, 4, names, 8) - expected: returns 0, because there is a match at the beginning

(names, 8, names, 2) - expected: returns 0, because there is a match at the beginning

(names, 8, namesSubset, 3) - expected: returns 2, because there is a match starting at position 2

(names, 8, namesSubset, 5) - expected: returns 2, because the "Jeff" at the end of the subsequence is capitalized

**int separate(string a[], int n, string separator);**

(names, -10, "") - expected: returns -1, because the size is negative

(names, 0, "") - expected: returns 0, because there are no elements smaller than splitter

(names, 4, "m") - expected: returns 3, because the first three are less than the splitter

(names, 8, "m") - expected: returns 5, because the first 6 elements are less than splitter

(names, 8, "z") - expected: returns 8, all of the elements are less than the splitter

(names, 8, "A") - expected: returns 0, because all of the elements are greater than the splitter

(names, 0, "A") - expected: returns 0, because there are no elements, but no invalid element is being accessed

(numberList, 15, "3") - expected: returns 4, because there are four elements that are less than three

(numberList, 15, "9") - expected: returns 15, because "9" is bigger than all other elements

(numberList, 15, "0") - expected: returns 0, because "0" is less than all