Maeneka Grewal -204901948

CS31

Project 4 Report

a. My first notable obstacle was attempting to write some of the functions without creating additional arrays. I couldn’t find a way to do it, until I eventually realized I could swap elements of the array. However, I still couldn’t figure out an efficient way to do this for split, and had to settle with creating a new array. Then I had to set this new array equal to the old array, using a for-loop. My second notable obstacle was running the program on g31. I had been running the program on g31 periodically to make sure everything compiled and that I had no out of bounds error. Up until the very last run, everything was fine. For some reason, at the end, countRuns and subsequence started returning the strangest “address sanitizer” error message when ran. I still can’t figure it out and I am very scared. I will overcome this obstacle by going to office errors, figuring out what this error message means, and then fixing it.

b.

|  |  |
| --- | --- |
| Test | Reason |
| String yeet[4] = {“mark”, “taylor”, “abbie”, “carl”};  assert(lookup(yeet, 4, “taylor”) == 1); | A normal test situation for the lookup function |
| String yeet[4] = {“mark”, “taylor”, “abbie”, “carl”};  assert(lookup(yeet, 3, “taylor”) == -1); | A test situation for when the target is not in the array |
| String yeet[4] = {“mark”, “taylor”, “abbie”, “carl”};  assert(appendToAll(yeet, 4, “@#”) == 4 && yeet[0] == “mark@#” && yeet[3] == “carl@#”); | A test situation for a normal appendage to this array |
| String yeet[4] = {“mark”, “taylor”, “abbie”, “carl”};  assert(appendToAll(yeet, -3, “@#”) == -1 && yeet[0] == “mark” && yeet[3] == “carl”); | A test situation for when the array size is less than 0 |
| String yeet[7] = {“mark”, “taylor”, “abbie”, “carl”, “umbrella”, “umbrella”, “giraffe”};  assert(positionOfMax(yeet, 7 ) == 4); | A test situation for when there is more than one ‘max’ object in the array |
| String yeet[0];  assert(positionOfMax(yeet, 0 ) == -1); | A test situation for when there are no elements in the array |
| String yeet[7] = {“mark”, “taylor”, “abbie”, “carl”, “umbrella”, “umbrella”, “giraffe”};  assert(rotateLeft(yeet, 7, 1) == 1 && yeet[1] == “abbie” && yeet[6] == “taylor”); | A test situation for a standard rotation of this array |
| String yeet[7] = {“mark”, “taylor”, “abbie”, “carl”, “umbrella”, “umbrella”, “giraffe”};  assert(rotateLeft(yeet, 0, 1) == -1 && yeet[1] == “taylor” && yeet[6] == “giraffe”); | A test situation for when the function will examine no elements of the array, so it must return -1 |
| String yeet[7] = {“mark”, “mark”, “abbie”, “carl”, “umbrella”, “umbrella”, “giraffe”};  assert(countRuns(yeet, 7) == 5); | A normal test situation for this function, with 5 separate elements. |
| String yeet[7] = {“mark”, “mark”, “abbie”, “carl”, “umbrella”, “umbrella”, “giraffe”};  assert(countRuns(yeet, 0) == 0); | A test situation where there are no elements in the array, so the function returns 0. |
| String yeet[7] = {“mark”, “mark”, “abbie”, “carl”, “umbrella”, “umbrella”, “giraffe”};  assert(countRuns(yeet, -3) == -1); | A test situation when the array size is less than 0, so the function returns -1 |
| String yeet[7] = {“mark”, “mark”, “abbie”, “carl”, “umbrella”, “umbrella”, “giraffe”};  assert(flip(yeet, 7) == 7 && yeet[0] == “giraffe” && yeet[6] == “mark” && yeet[4] == “abbie”); | A test situation flipping all the elements in this array |
| String yeet[7] = {“mark”, “mark”, “abbie”, “carl”, “umbrella”, “umbrella”, “giraffe”};  assert(flip(yeet, 4) == 7 && yeet[0] == “umbrella” && yeet[3] == “mark”); | A test situation flipping only a certain portion of this array. |
| String yeet[7] = {“mark”, “mark”, “abbie”, “carl”, “umbrella”, “umbrella”, “giraffe”};  String yay[5] = {“mark”, “mark”, “abbie”, “”, “carl”};  assert(differ(yeet, 7, yay, 5) == 3); | A normal test situation for finding the differing position in two arrays. |
| String yeet[7] = {“mark”, “mark”, “abbie”, “carl”, “umbrella”, “umbrella”, “giraffe”};  String yay[5] = {“mark”, “mark”, “abbie”, “”, “carl”};  assert(differ(yeet, 7, yay, 2) == 2); | A test situation where the arrays are equal until one array runs out, so the smaller array size is returned. |
| String yeet[7] = {“mark”, “mark”, “abbie”, “carl”, “umbrella”, “umbrella”, “giraffe”};  String yay[3] = {“carl”, “umbrella”, “umbrella”};  assert(subsequence(yeet, 7, yay, 3) == 3); | A normal test situation where the second array is found in the first, so the position where the subsequence begins is returned. |
| String yeet[7] = {“mark”, “mark”, “abbie”, “carl”, “umbrella”, “umbrella”, “giraffe”};  String yay[3] = {“carl”, “hello”, “umbrella”};  assert(subsequence(yeet, 7, yay, 3) == -1); | A test situation where the second array is not found in the first, so -1 is returned. |
| String yeet[7] = {“mark”, “mark”, “abbie”, “carl”, “umbrella”, “umbrella”, “giraffe”};  String yay[3] = {“carl”, “umbrella”, “umbrella”};  assert(subsequence(yeet, 7, yay, 0) == 0); | A test situation where the second array is of size 0, therefore 0 is returned. |
| String yeet[7] = {“mark”, “mark”, “abbie”, “carl”, “umbrella”, “umbrella”, “giraffe”};  String yay[5] = {“carl”, “umbrella”, “”, “abbie”, “cheese”};  assert(lookupAny (yeet, 7, yay, 5) == 2); | A test situation where an element of the first array is found in the second, so the first position is returned. |
| String yeet[7] = {“mark”, “mark”, “abbie”, “carl”, “umbrella”, “umbrella”, “giraffe”};  String yay[5] = {“turtle”, “hamster”, “”, “hippo”, “cheese”};  assert(lookupAny (yeet, 7, yay, 5) == -1); | A test situation where elements of the first array are not found in the second, so -1 is returned. |
| String yay[5] = {“turtle”, “hamster”, “computer”, “hippo”, “cheese”};  assert(split (yay, 5, “grass”) == 2 && yay[0] == “computer” && yay[1] == “cheese”); | A test situation where the string splitter is not in the array, and the position of the first string greater than splitter is returned. |
| String yay[5] = {“turtle”, “hamster”, “computer”, “hippo”, “cheese”};  assert(split (yay, 5, “hippo”) == 3); | A test situation where the splitter itself is in the array, so the position of the splitter in the new array is returned. |
| String yay[5] = {“turtle”, “hamster”, “computer”, “hippo”, “cheese”};  assert(split (yay, 5, “zamboni”) == 5); | A test situation where the splitter is greater than all of the elements in the array, so the size of the array is returned. |