Kevin Wang

UID: 205209507

Project 4 Report

Notable obstacles:

* shiftLeft function took some time to figure out as the way I implemented my function doesn’t actually move all the elements over by the amount given, but rather it shifts the elements that can be moved to the position where it would’ve been shifted and fills out the rest of the spots that would have been affected.
* The divide function I initially implemented it in a way that required an initialization of an empty array of an arbitrary size (about 1,000) to add in elements smaller than divider and I rearranged the original array based off the values that were smaller than divider and larger than divider. The new method I changed the function to does not require initialization of another array and instead jus swaps out the larger values to the end of the array.

Test Data

locateMaximum:

* string max[5] = { “c”, “a”, “b”, “c”, “c”}
  + assert(locateMaximum(people, 5) == 0);
    - Check when maximum is the first element of the array
* string max[5] = { “a”, “a”, “b”, “c”, “c”}
  + assert(locateMaximum(people, 3) == 2);
    - check that changing the ‘n’ value gets the correct maximum
  + assert(locateMaximum(people, 5) == 3);
    - returns the first position of max when there are two of the same value

hasDuplicates:

* string dup[10] = { "samwell", "jon", "margaery", "daenerys", "a", "a", "a", "a", "e", "e" };
  + assert(hasDuplicates(dup, 10) == true);
    - checks that it correctly identifies duplicates

matchingValuesTogether:

* string dup[10] = { "samwell", "jon", "margaery", "daenerys", "a", "a", "a", "a", "e", "e" };
  + assert(matchingValuesTogether(dup, 10) == true);
    - checks that it correctly identifies matching values
* string dup[10] = { "samwell", "jon", "margaery", "daenerys", "a", "a", "b", "a", "e", "e" };
  + assert(matchingValuesTogether(dup, 10) == false);
    - checks that when there is a separation between duplicates it correctly identifies as false

countSs:

* string sS[5] = { "ssss", "g", "s", "sss", "SSS" };
  + assert(countSs(sS, 5) == 11);
    - correctly counts all of the ‘s’ and ‘S’ within all the elements of the array

isInDecreasingOrder:

* string dec[5] = { "c", "b", "a", "AA", "A" };
  + assert(isInDecreasingOrder(dec, 5) == true);
    - verifies that each element is indeed decreasing

divide:

* string div[5] = { "a", "c", "e", "h", "e" };
  + assert(divide(div, 5, "e") == 4 );
    - finds position of first element greater than divider
* string div[5] = { "a", "c", "e", "e", "e" };
  + assert(divide(div, 5, "e") == 0 );
    - returns 0 if there are no elements that are greater than divider
* string div[5] = { "b", "c", "e", "e", "e" };
  + assert(divide(div, 5, "a") == 0 );
    - return position 0 if all elements of the array are greater than divider
* string div[5] = { "a", "c", "a", "e", "e" };
  + assert(divide(div, 3, "b") == 2 );
    - correctly returns position when n is changed

shiftLeft:

* string shift[10] = { "samwell", "jon", "margaery", "daenerys", "tyrion", "a", "b", "c", "d", "e" };
  + assert(shiftLeft(shift, 8, 5, "lol") == 5 && shift[5] == "lol" && shift[0] == "a" && shift[8] == "d");
    - verify correct number of placeholders, position of “a” after it is shifted, and that “d” is still in the correct position because it was not moved due to size of n being 8