Project 4 Report

1. Notable obstacles
2. Looking at each character in a string that’s part of an array
   1. At first, I tried using things like isdigit(array[i]) or isupper(array[i]). However, this did not work because these commands are only for characters in a string, not a string as a whole. To overcome this, I made a string that is the current array element. I then looped through each character in the string and used those commands there. For example, in hasNoCapitals:

**for** (**int** i=0; i<n-1; i++) { //go through array elements

string s = array[i];

**for** (size\_t j=0; j < s.size()-1; j++) { //go through string

**if** (isupper(s.at(j))) { //check if capital letter

**return**(**false**);

}

}

1. Replacing only the first and last occurrences of a character
   1. At first, I would change all occurrences instead of all because I was unsure how to find the last position. I first tried counting backwards as well, but I still couldn’t control when the replacement happens. I solved this by first going into the string and counting the number of occurrences. Then, I would go through the string again with a new counter. If this new counter was equal to or less than 2, I could just replace all occurrences. If there were occurrences in the middle of the string, I would replace only the first one and when the new counter equals total number of occurrences.
2. Test data

//strings to test code

string a[6] = { "123", "456", "789", "gamma", "beta", "delta" }; //given ex

string num[3] = {"124", "-3", "0.1"}; //neg sign bad, leading 0 ok

string data[5] = { "mamaBbcca", "mamaBbcca", "12,", "98.76", "tyrion" }; //given ex

string folks[8] = { "samwell", "jon", "margaery", "daenerys", "tyrion", "sansa", "magdalena", "jon" }; //given ex

string simple[5] = { "0", "1.0", "-2", ".", "."}; //easier to see locate max

string repeat[6] = {"ate", "abe" "age" "are" "ate" "ate"};

string c[4] = {"hearts" "hearts" "hearts" "clover"}; //set of 3 matching values

string s[6] = {"s" "s" "s" "t" "t" "s"}; //3 in a row, 2 in a row, 1

string d[3] = {"." "1" "hi"}; //optional decimal point

//asserts to test code

assert(locateMaximum(data, 0) == -1); //n is 0

assert(locateMaximum(data, -100) == -1); //n is less than 0

assert(locateMaximum(s, 4) == 0); //smallest index with max calue, next to each other

assert(locateMaximum(repeat, 6) == 0); //smallest index that has max value, spread out

assert(locateMaximum(data, 5) == 4); //given ex

assert(countFloatingPointValues(a, 0) == -1); //n is 0

assert(countFloatingPointValues(a, -10) == -1); //n is less than 0

assert(countFloatingPointValues(data, 5) == 1); //given ex

assert(countFloatingPointValues(simple, 3) == 2); //optional decimal ok, negative sign not allowed

assert(countFloatingPointValues(simple, 4) == 3); //n is less than 0

assert(hasNoCapitals(data, 0) == **true**); //n is 0

assert(hasNoCapitals(data, -300) == **true**); //n is less than 0

assert(hasNoCapitals(folks, 8) == **true**); //does not have capital letters

assert(hasNoCapitals(data, 5) == **false**); //has capital letters

assert(identicalValuesTogether(data, 0) == **false**); //n is 0

assert(identicalValuesTogether(data, -50) == **false**); //n is less than 0

assert(identicalValuesTogether(data, 5) == **true**); //two next to each other

assert(identicalValuesTogether(folks, 8) == **false**); //repeated not next to each other

assert(identicalValuesTogether(c,4) == **true**); //3 in a row

assert(identicalValuesTogether(s,5) == **true**); //3 in a row, 2 in a row

// assert(identicalValuesTogether(s,6) == false); //3 in a row, 2 in a row

🡪This one returns true instead of false The string has 3 s’s together and 2 t’s together, but 1 s by itself. Could not figure out how to check further after getting true.

assert(hasTwoOrMoreDuplicates(c, 0) == **false**); //n is 0

assert(hasTwoOrMoreDuplicates(c, -8) == **false**); //n is less than 0

assert(hasTwoOrMoreDuplicates(folks, 8) == **false**); //given ex, one set of 2, not two

assert(hasTwoOrMoreDuplicates(folks, 3) == **false**); //given ex, no 2 sets of 2 or 1 set of 3

assert(hasTwoOrMoreDuplicates(c, 4) == **true**); //1 set of 3

assert(hasTwoOrMoreDuplicates(c, 4) == **true**); //1 set of 3

assert(hasTwoOrMoreDuplicates(s, 6) == **true**); //2 sets of 2

assert(shiftLeft(data, 0, 2, "!") == -1); //n is 0

assert(shiftLeft(data, -5000, 2, "!") == -1); //n is less than 0

assert(shiftLeft(data, 5, 2, "foo") == 2); //given ex, amount<n

assert(shiftLeft(data, 5, 10, "bar") == 5); //given ex, amount>n

assert(shiftLeft(data, 5, -5, "foobar") == -1); //given ex, amount less than 0

assert(replaceFirstAndLastOccurrences(folks, 0, 'A', 'Z') == -1); //n is 0

assert(replaceFirstAndLastOccurrences(folks, -2, 'A', 'Z') == -1); //n is less than 0

assert(replaceFirstAndLastOccurrences(folks, 8, 'A', 'Z') == 0); //given ex, no A's

assert(replaceFirstAndLastOccurrences(folks, 8, 'a', 'Z') == 8); //given ex, some with occurrences in middle