Name: Sidharth Sudhir

UID: 805 759 182

CS31 Project 4 Report

Obstacle # 1: Indexing out of list in several problems

Similar to mistakes that I made in past projects while indexing strings, for the first 3 functions I indexed beyond the list length. To fix this, I changed the [i+1] in the index to [i-1] and made other modifications to the program.

Obstacle # 2: Edge cases

For each function that was coded, we had to deal with negative and 0 input cases. This was confusing as some of the functions required that we dealt with n = 0 while others did not. Figuring this out required writing and hand tracing some of the programs out on paper.

Test Cases

Description of test cases: Each of the groups of test cases dealt with 2-3 standard cases with regular inputs and 1-2 of the edge case handling that were made for.

**Function:** *reduplicate*

string test\_arr1[6] = { "mahi", "bon", "cous", "", "tar", "mur" };

* assert(reduplicate(test\_arr1, 6) == 6 && test\_arr1[1] == "bonbon" && test\_arr1[3] == "");

string c1[4] = { "ma", "can", "tu", "do" };

* assert(reduplicate(c1, 4) == 4 && c1[0] == "mama" && c1[3] == "dodo");

string c2[4] = { "ma", "can", "tu", "do" };

* assert(reduplicate(c2, 0) == -1 && c2[0] == "ma" && c2[3] == "do");

string c3[4] = { "ma", "can", "tu", "do" };

* assert(reduplicate(c3, -2) == -1 && c3[0] == "ma" && c3[3] == "do");
* assert(reduplicate(test\_arr1, 0) == -1);

**Function:** *locate*

string h[7] = { "moana", "mulan", "ariel", "tiana", "", "belle", "elsa" };

* assert(locate(h, 7, "belle") == 5);
* assert(locate(h, 7, "ariel") == 2);
* assert(locate(h, 2, "ariel") == -1);

string emptyarr[1] = {};

* assert(locate(emptyarr, 0, "hello") == -1);

**Function:** *locationOfMax*

string test\_arr3[6] = { "elsa", "ariel", "mulan", "tiana", "belle", "moana" };

string empty\_arr3[1] = {};

* assert(locationOfMax(test\_arr3, 6) == 3);
* assert(locationOfMax(h, 7) == 3);
* assert(locationOfMax(empty\_arr3, 0) == -1);

**Function:** *circleLeft*

string g1[4] = { "moana", "mulan", "belle", "raya" };

* assert(circleLeft(g1, 4, 1) == 1 && g1[1] == "belle" && g1[3] == "mulan");

string g2[4] = { "moana", "mulan", "belle", "raya" };

* assert(circleLeft(g2, -1, 1) == -1 && g2[1] == "mulan" && g2[3] == "raya");

string g3[4] = { "moana", "mulan", "belle", "raya" };

* assert(circleLeft(g3, 4, 5) == -1 && g2[1] == "mulan" && g2[3] == "raya");

**Function:** *enumerateRuns*

string d[5] = {"mulan", "mulan", "mulan", "belle", "belle"};

* assert(enumerateRuns(d, 5) == 2);

string a[10] = {“a”, “a”, “b”, “b”, “c”, “a”, “a”, “d”, “b”, “b”}

* assert(enumerateRuns(a, 10) == 6);

string diseases[6] = {“A”, “A”, “S”, “C”, “C”, “A”};

* assert(enumerateRuns(diseases, 4) == 3);
* assert(enumerateRuns(diseases, -1) == -1);
* assert(enumerateRuns(diseases, 0) == 0);

**Function:** *flip*

string f[3] = { "tiana", "ariel", "raya" };

* assert(flip(f, 3) == 3);
* assert(flip(f, -4) == -1);

string letters[8] = {“a”, “b”, “c”, “d”, “e”, “f”, “g”, “h”};

* asster(flip(letters, 4) == 4);

string nullList[1] = {};

* assert(flip(nullList, -1) == -1);

**Function:** *locateDifference*

string h[7] = { "moana", "mulan", "ariel", "tiana", "", "belle", "elsa" };

string g[4] = { "moana", "mulan", "belle", "raya" };

* assert(locateDifference(h, 4, g, 4) == 2);
* assert(locateDifference(h, 2, g, 1) == 1);
* assert(locateDifference(h, 2, g, -56) == -1);
* assert(locateDifference(h, 2, g, 4) == -1);

**Function:** *subsequence*

string names[10] = { "moana", "mulan", "raya", "tiana", "merida" };

string names1[10] = { "mulan", "raya", "tiana" };

string names2[10] = { "moana", "tiana" };

string e[4] = { "ariel", "tiana", "", "belle" };

string single\_arr[1] = { "belle" };

string empty[1] = {};

* assert(subsequence(names, 5, names1, 3) == 1); // returns 1
* assert(subsequence(names, 4, names2, 2) == -1); // returns -1
* assert(subsequence(h, 7, e, 4) == 2);
* assert(subsequence(h, 7, empty, 0) == -1);
* assert(subsequence(h, 7, single\_arr, 1) == 5);

**Function:** *locateAny*

string f[3] = { "tiana", "ariel", "raya" };

* assert(locateAny(h, 7, f, 3) == 2);

string f1[1] = {};

* assert(locateAny(h, 7, f1, 0) == -1);

string f2[10] = { "elsa", "merida", "tiana", "mulan" };

* int v = locateAny(names, 6, f2, 4); // returns 1 (a1 has "mulan" there)

string f3[10] = { "belle", "ariel" };

* int w = locateAny(names, 6, f3, 2); // returns -1 (a1 has none)

**Function:** *separate*

string cast[6] = { "elsa", "ariel", "mulan", "belle", "tiana", "moana" };

* assert(separate(cast, 6, "merida") == 3);

string cast2[4] = { "mulan", "tiana", "ariel", "raya" };

* assert(separate(cast2, 4, "raya") == 2);
* assert(separate(h, 7, "elsa") == 3);
* assert(separate(empty, 0, "elsa") == -1);