const int m = 9;

string a[m] = { "thor", "romanoff", "parker", "parker", "stark", "stark", "stark", "parker", "parker" };

//cout << "Enter a list of words with a comma followed by each one: " << endl;

//getline(cin, a);

int n = 9;

// cout << "Enter the number out of the total list you want to check: " << endl;

//cin >> n; cin.ignore(10000, '\n');

string target;

//cout << Enter word you seek: " << endl; getline(cin, target);

int begin;

int end;

string people[5] = { "parker", "thor", "stark", "banner", "romanoff" };

cout << tally(a, n, "parker") << endl;

cout << findMatch(people, 5, "banner") << endl;

cout << findRun(a, n, "parker", begin, end) << endl;

cout << begin << ", " << end << endl;

cout << findRun(a, n, "romanoff", begin, end) << endl;

cout << begin << ", " << end << endl;

cout << findRun(a, n, "rogers", begin, end) << endl;

cout << begin << ", " << end << endl;

cout << positionOfMin(a, n) << endl;

1. One challenge included keeping track of proper sizes for arrays and checking if an array had that given element. Another was making sure I didn’t have undefined behavior. Another was checking my bounds for the loops.
2. **Bool Subsequence:**

string big[10] = { "parker", "thor", "stark", "banner", "romanoff", "stark" };

string little1[10] = { "thor", "banner", "romanoff" };string little2[10] = { "stark", "thor" }; string little3[10] = { "thor", "stark", "stark" }; string little4[10] = { "thor", "thor", "stark" };

Cases:

cout << subsequence(big, 6, little1, 3) << endl; (simple case from spec)

cout << subsequence(big, 6, little2, 2) << endl; (case when subsequence had elements in non-same order)

cout << subsequence(big, 6, little3, 3) << endl; (case when subsequence has non unique elements)

cout << subsequence(big, 6, little4, 3) << endl; (same case as before, but with a different order)

string h[7] = { "romanoff", "thor", "rogers", "banner", "", "parker", "rogers" };

string g[4] = { "romanoff", "thor", "banner", "parker" };

Cases: subsequence(h, 7, g, 4) (case where larger string has an empty string)

string a[9] = { "thor", "romanoff", "parker", "parker", "stark", "stark", "stark", "parker", "parker" };

string people[5] = { "parker", "thor", "stark", "banner", "romanoff" };

int n=9;

string f[6] = { "rhodes", "banner", "stark", "parker", "thor", "rogers" };

string g[4] = { "romanoff", "rogers", "thor", "banner" };

string x2[4] = { "rhodes", "rhodes", "tchalla", "thor" };

string y2[4] = { "banner", "parker", "rhodes", "rogers" };

string zed2[10];

string x[5] = { "banner", "rhodes", "rogers", "stark", "tchalla" };

string y[4] = { "parker", "rogers", "rogers", "thor" };

string zed[20];

string cast[5] = { "parker", "thor", "stark", "banner", "romanoff" };

string roles[4] = { "parker", "thor", "tchalla", "rhodes" };

string e[5] = { "parker", "parker", "parker", "thor", "thor" };

string f[6] = { "rhodes", "banner", "stark", "parker", "thor", "rogers" };

string g[4] = { "romanoff", "rogers", "thor", "banner" };

string gee[4] = { "romanoff", "thor", "banner", "parker" };

**Tally:**

cout << tally(a, n, "parker") << endl; (simpler case)

**Find Match:**

cout << findMatch(people, 5, "banner") << endl; (simpler case)

**Find Run:**

cout << findRun(a, n, "parker", begin, end) << endl; (simpler case)

**Position of Min:**

cout << positionOfMin(a, n) << endl; (simpler case)

Move to End:

moveToEnd(people, 5, 1); (simpler case)

Move to Begin:

moveToBeginning(people, 5, 2) << endl; (simpler case)

findDifference:

cout << findDifference(cast, 5, roles, 4) << endl; (both were examples from spec)

cout << findDifference(cast, 2, roles, 1) << endl;

EliminateDup:

cout << eliminateDups(a, 9) << endl;

cout << eliminateDups(e, 5) << endl; (examples from spec)

makeMerge: cout << makeMerger(x, 5, y, 4, zed, 20) << endl;

(example from spec)

Separate:

cout << separate(f, 6, "romanoff") << endl;

cout << separate(g, 4, "rogers") << endl;

cout << separate(h, 7, "rogers") << endl;

(all examples from spec)

Tests failed: subsequence(h, 7, gee, 4) << endl;

(works on g31 but fails on Visual Studio)