Calvin Liu

804182525

Report

1.For this project, there were not a lot of notable obstacles where I had no idea what to do. The main point was to manipulate the array using other functions that we already learned. The only hard part was figuring out the concept of how to solve the problems using algorithms and what approach you can use. For example, for the flip function, you had to be able to switch the variables by copying them into a dummy variable; then you needed to create an algorithm where the counter variable was increasing, but then you needed to have another variable decreasing. As I went down the functions, the functions got more and more difficult.

2.Test cases used:

int main()

{

string h[7] = { "peter", "lois", "meg", "chris", "", "stewie", "meg" };

string a[4] = { "a", "b", "c", "d"};

string b[4] = { "", "b", "z", "z"};

string c[4] = { "a", "", "b", "c"};

string d[5] = { "asdf", "asdf", "asdf", "asdf", "asdf"};

assert(itIsIncreasingOrder(a, 4));

assert(itIsIncreasingOrder(b, 3));

assert(!itIsIncreasingOrder(c, 3));

assert(tally(h, 7, "meg") == 2);

assert(tally(h, 7, "") == 1);

assert(tally(h, 7, "quagmire") == 0);

assert(tally(h, 0, "meg") == 0);

assert(tally(d, 5, "asdf") == 5);

assert(findFirst(h, 7, "meg") == 2);

assert(findFirst(h, 2, "meg") == -1);

assert(findFirst(c, 3, "") == 1);

assert(findFirst(a, 4, "d") == 3);

assert(findFirst(a, 0, "d") == -1);

int bg;

int en;

string i[7] = {"monica", "chandler", "ross", "rachel", "joey", "joey", ""};

assert(findFirstSequence(h, 7, "chris", bg, en) && bg == 3 && en == 3);

assert(findFirstSequence(i, 7, "joey", bg, en) && bg == 4 && en == 5);

assert(!findFirstSequence(i, 0, "pheobe", bg, en));

string g[4] = { "peter", "lois", "chris", "stewie" };

string l[5] = { "qwer", "asdf", "asdf", "qwer", "qwer"};

string t[6] = { "qwer", "asdf", "asdf", "qwer", "exe", "qwer"};

string k[4] = { "", "king", "kid", "prize"};

string u[4] = { "", "kid", "king", "prize"};

string m[5] = { "", "king", "kid", "prize", "many"};

string n[5] = { "", "king", "kid", "balloon", "many"};

string p[0];

string q[1] = {"onething"};

assert(positionOfMin(g, 4) == 2);

assert(positionOfMin(g, 0) == -1);

assert(positionOfMin(l, 5) == 1);

assert(positionOfMin(l, 3) == 1);

assert(positionOfMin(l, 1) == 0);

assert(positionOfMin(k, 4) == 0);

assert(disagree(h, 4, g, 4) == 2);

assert(disagree(k, 4, m, 5) == 3);

assert(disagree(n, 5, m, 5) == 3);

assert(disagree(n, 1, m, 5) == 0);

assert(disagree(n, 2, m, 5) == 1);

assert(subsequence(h, 7, g, 4));

assert(subsequence(n, 5, p, 0));

assert(subsequence(m, 5, n, 3));

assert(!subsequence(m, 5, n, 4));

assert(moveToEnd(g, 4, 1) == 1 && g[1] == "chris" && g[3] == "lois");

assert(moveToEnd(n, 5, 3) == 3 && n[4] == "balloon" && n[0] == "" && n[3] == "many");

assert(moveToEnd(p, 0, 0) == -1);

assert(moveToEnd(q, 1, 0) == 0 && q[0] == "onething");

assert(moveToEnd(t, 6, 4) == 4 && t[5] == "exe" && t[4] == "qwer" && t[3] == "qwer" && t[2] == "asdf" && t[1] == "asdf" && t[0] == "qwer");

string f[4] = { "chris", "stewie", "meg", "lois" };

assert(moveToBeginning(f, 4, 2) == 2 && f[0] == "meg" && f[2] == "stewie");

assert(moveToBeginning(g, 4, 1) == 1 && g[0] == "lois" && g[1] == "peter");

assert(moveToBeginning(m, 5, 2) == 2 && m[0] == "kid" && m[2] == "king");

assert(moveToBeginning(m, 1, 0) == 0);

assert(moveToBeginning(i, 7, 3) == 3 && i[3] == "ross" && i[2] == "chandler");

string e[5] = { "chris", "chris", "chris", "meg", "meg" };

assert(removeDups(e, 5) == 2 && e[1] == "meg");

assert(removeDups(d, 5) == 1 && d[0] == "asdf");

assert(removeDups(l, 5) == 2 && l[0] == "qwer" && l[1] == "asdf");

assert(removeDups(l,6) == 3 && l[0] == "qwer" && l[1] == "asdf" && l[2] == "exe");

assert(removeDups(l, 0) == 0);

string x[4] = { "brian", "lois", "lois", "quagmire" };

string y[4] = { "chris", "lois", "meg", "stewie" };

string z[10];

assert(mingle(x, 4, y, 4, z, 10) == 8 && z[5] == "meg");

assert(mingle(e, 5, k, 4, z, 6) == -1);

assert(mingle(e, 5, k, 4, z, 10) == -1);

assert(mingle(e, 5, u, 4, z, 10) == 9 && z[4] == "kid" && z[1] == "chris");

assert(divide(h, 7, "meg") == 3);

assert(divide(m, 5, "z") == 5);

assert(divide(n, 5, "a") == 1);

assert(divide(y, 4, "o") == 3);

assert(divide(y, 0, "o") == 0);

assert(divide(y, -1, "o") == -1);

cout << "All tests succeeded" << endl;

}