Ethan Wong

CS31

Professor Smallberg

305319001

Project 4 Report

1. My main obstacle with creating this program was the “separate” function. I tried to implement different strategies to get the code to work, but they were all overly complicated and did not execute as expected. After thinking about what to do to solve the issue, I decided it would be best to implement the “rotateLeft” function to help with the separation of the strings in the array. Doing this along with creating some additional variables to relocate the position of the separator in the array (if necessary) allowed me to create a function that worked as I wanted.

string clan[6] = { "clay", "peter", "ree", "tones", "dina", "bryson" };

string gang2[5] = { "ree", "suezette", "peter", “suezette”, "stella" };

string d[9] = {"tones", "bryson", "stella", "stella", "dina", "dina", "dina", "stella", "stella"};

string friends[6] = { "bryson", "stella", "", "tones", "suezette", "clay" };

string squad[5] = { "bryson", "stella", "clay", "", "tones" };

string names[10] = { "logan", "ree", "suezette", "selma", "bryson", "peter", “ree”, “selma”, “suezette” };

string Xnames[10] = { "ree", "suezette", "selma" };

string Ynames[10] = { "logan", "selma" };

appendToAll

* appendToAll(clan, 6, “!!!”) - appends something to all parts of array
* appendToAll(clan, 0, “!!!”) - appends something to no parts of the array
* appendToAll(clan, 3, “!!!”) - appends something to part of the array
* appendToAll(clan, -4, “!!!”) - tests negative value for position in array

lookup

* lookup(clan, 6, “bryson”) - finds value in array
* lookup(gang2, 5, “suezette”) - finds value in array that appears twice
* lookup(clan, 6, “brady”) - looking up element not in the array
* lookup(clan, 0, “clay”) - looking up in no part of the array
* lookup(clan, -6, “clay”) - testing negative value for position in array

positionOfMax

* positionOfMax(clan, 6) - finding largest value in array
* positionOfMax(clan, 4) - finding largest value in part of the array
* positionOfMax(gang2, 5) - looking for largest value in array if it appears more twice
* positionOfMax(clan, 0) - looks in no part of the array
* positionOfMax(clan, -5) - testing negative value for position in array

rotateLeft

* rotateLeft(clan, 6, 0) - rotates all elements left
* rotateLeft(clan, 6, 5) - rotates end elements to the left
* rotateLeft(clan, 6, 6) - position is equal to n
* rotateLeft(clan, -3, 4) - testing for when value for n is less than 0
* rotateLeft(clan, 6, -2) - tests negative position

countRuns

* countRuns(d, 9) - counts amount of sequences of consecutive elements
* countRuns(d, 6) - counts amount of sequences of consecutive elements in part of array
* countRuns(d, 0) - counts amount of sequences of consecutive elements in no parts of array
* countRuns(d, -4) - testing negative value for position in array

flip

* flip(clan, 6) - performs flip function on entire array
* flip(clan, 0) - flips no part of the array
* flip(clan, 5) - flips portion of the array
* flip(gang2, 5) - flip an odd amount of elements of the array
* flip(clan, -5) - testing negative value for position in array

differ

* differ(friends, 6, squad, 5) - compares difference with two arrays
* differ(friends, 0, squad, 1) - tests when one of the n values is 0
* differ(friends, 0, squad, 0) - tests when both of the n values are 0
* differ(friends, 1, squad, -2) - testing for when value(s) for n is/are less than 0
* differ(friends, 2, squad, 1) - test when there is no difference

subsequence

* subsequence(names, 6, Xnames, 3) - tests subsequence existing in the two arrays
* subsequence(names, 9, Xnames, 3) - there are two cases of subsequence
* subsequence(names, 5, Ynames, 2; - no subsequence exists
* subsequence(names, 0, Xnames, 3) - logic error when n1 < n2
* subsequence(names, 2, Xnames, 0) - tests what happens when one of the n values is 0
* subsequence(names, 0, Xnames, 0) - tests what happens when both n values are 0
* subsequence(names, -4, Xnames, -3) - testing for when value(s) for n is/are less than 0

lookupAny

* lookupAny(clan, 6, gang2, 5) - contains any of the strings
* lookupAny(clan, 0, heroes2, 2) - tests what happens when one of the n values is 0
* lookupAny(clan, -2, heroes2, 4) - testing for when value(s) for n is/are less than 0
* lookupAny(clan, 6,Ynames, 2) - does not contain any of the strings

separate

* separate(clan, 6, “dina”) - separating an array when the separator is in the array
* separate(clan, 6, "logan") - separating an array when the separator isn’t in the array
* separate(clan, 0, “dina”) - testing for when value for n is 0
* separate(clan, -2, “dina”) - testing for when value for n is less than 0