Alex Wasdahl

CS31 Spring 2022

The two most notable obstacles I faced were:

1. Debugging an error in moveToEnd and moveToBeginning in which I had referenced pos instead of i, which led to only one item of the rest of the array being pushed to replace the one that had been moved. Easy fix, but took a long time
2. Figuring out, in the makeMerger function, how to keep track of the array elements from a1 and a2 that have already been added to the result array to avoid duplicateion

List of test data:

String colors[7] = {“orange”, “red”, “indigo”, “blue,” “”, “green,” “indigo”}

1. enumerate
   1. assert(enumerate(colors, -99, "indigo") == -1); Passing in -99 for n to confirm function returns -1
   2. assert(enumerate(colors, 7, "indigo") == 2); tests to make sure both instances are counted
   3. assert(enumerate(colors, 7, "") == 1); test if recognizes empty string
   4. assert(enumerate(colors, 7, "apple") == 0); test string not in array
2. findMatch
   1. assert(findMatch(colors, 7, "indigo") == 2); test if finds both instances
   2. assert(findMatch(colors, 6, “indigo”) == 1); test if finds only one instance due to second instance being out of range of n
   3. assert(findMatch(colors, -1, indigo”) == -1); test if negative n properly returns -1
3. findRun

int b; int e;

* 1. assert(findRun(colors, 7, “blue”, b, e) && b == 3 && e == 3; test if function properly returns true and begin and end parameters are correct

string fruits[4] = (“mango”, “strawberry”, “apple”, “elderberry”}

string morefruits[4] = {“mango”, “strawberry”, “apple”, “durien”}

1. findMin
   1. assert(findMin(fruits, 4) == 2); test if function properly returns apple as smallest string
2. moveToEnd
   1. assert(moveToEnd(fruits, 4, 1) == 1 && fruits[1] == “apple” && fruits[3] == “strawberry”); test if moveToEnd properly returns the position being moved, the string is properly moved to the end of the array, and the other items are pushed back properly
3. moveToBeginning
   1. assert(moveToBeginning(fruits, 4, 2) == 1 && fruits[2] == “strawberry” && fruits[0] == “apple”); test if moveToBeginning properly returns the position being moved, the string is properly moved to the beginning of the array, and the other items are pushed forward properly
4. findDifference
   1. assert(findDifference(fruits, 4, morefruits, 4) == 3; test if function identifies index 3 as the proper index at which the difference occurs

string evenmorefruits[5] = {“kiwi”, “kiwi”, “kiwi”, “watermelon”, “watermelon”}

1. removeDups
   1. assert(removeDups(evenmorefruits, 5) == 2 && evenmorefruits[1] == “watermelon”; tests if duplicates are removed and two unique items are returned, with the second one being “watermelon”

string morecolors[3] = {“red”, “blue”, “indigo”}

1. subsequence
   1. assert(subsequence(colors, 7, morecolors, 3); tests if finds all items from morecolors in sequence in colors and returns true if so
   2. assert(subsequence(morecolors, 3, colors, 7); tests if returns false (7 > 3)

string a[4] = { "kennedy", "kennedy", "truman", "trump" };  
string b[4] = { "bush", "clinton", "kennedy", "obama" };  
string c[10];

1. makeMerger
   1. assert(makeMerger(a, 4, b, 4, c, 10) == 8 && c[5] == "obama"); tests if returns proper number of elements (8) and the fifth element (obama) is positioned in the alphabetically correct spot
2. divide
   1. assert(divide(colors, 7, “indigo”) ==3);