1. I had some trouble coming up with a way to implement the subsequence function, since I could not figure out a way to indicate the start of a sequence and how to start checking for the consecutive sequence after that. I ended up using a boolean value to indicate the beginning of a possible sequence and initiate the check for the correctness of the consecutive sequence. I also had some bugs caused by accidentally checking one element more than the maximum size of the array, but I solved that problem by stopping the check at the maximum index.
2. appendToAll:

|  |  |
| --- | --- |
| Input | Reason |
| string cand1[5] = { "a", "A", "ab", "jill", "zjamu" };  int length1 = 5;  appendToAll(cand1, length1, "lol"); | Make sure the function does append strings to the end of every string in the array and returns length1 |
| string cand1[5] = { "a", "A", "ab", "jill", "zjamu" };  int length1 = -5;  appendToAll(cand1, length1, "lol"); | Function should return -1 when it received a negative array length |

lookup:

|  |  |
| --- | --- |
| Input | Reason |
| string cand1[5] = { "a", "A", "ab", "jill", "zjamu" };  int length1 = 5;  lookup(cand1, length1, "jill"); | Make sure the function returns the index of the target string |
| string cand1[5] = { "a", "A", "ab", "jill", "zjamu" };  int length1 = -5;  lookup(cand1, length1, "A"); | Function should return -1 when it received a negative array length |
| string cand1[5] = { "a", "A", "ab", "jill", "zjamu" };  int length1 = 5;  lookup(cand1, length1, "trump"); | Make sure the function returns -1 when the target string can’t be found in the array |

positionOfMax:

|  |  |
| --- | --- |
| Input | Reason |
| string cand1[5] = { "a", "A", "ab", "jill", "zjamu" };  int length1 = 5;  positionOfMax(cand1, length1); | Make sure the function returns the index with the string with the largest value |
| string cand1[5] = { "a", "A", "ab", "jill", "zjamu" };  int length1 = -5;  positionOfMax(cand1, length1); | Make sure the function returns -1 when it received a negative array length |
| string cand1[5] = { "a", "A", "ab", "jill", "zjamu" };  int length1 = 0;  positionOfMax(cand1, length1); | Make sure the function returns -1 when it received 0 for array length (no element in the array) |

rotateLeft:

|  |  |
| --- | --- |
| Input | Reason |
| string running[5] = { "evan", "donald", "gary", "jill", "hillary" };  int length1 = 5;  rotateLeft(running, length1, 1); | Make sure the function eliminates the item at position pos by copying all elements after it one place to the left and returns the original position of the item |
| string running[5] = { "evan", "donald", "gary", "jill", "hillary" };  int length1 = -5;  rotateLeft(running, length1, 1); | Make sure the function returns -1 when it received a negative array length |

countRuns:

|  |  |
| --- | --- |
| Input | Reason |
| string d[9] = { "tim", "ajamu", "mike", "mike", "donald", "donald", "donald", "mike", "mike" };  int length1 = 9;  countRuns(d, length1); | Make sure the function returns the number of sequences of one or more consecutive identical items in a |
| string d[9] = { "tim", "ajamu", "mike", "mike", "donald", "donald", "donald", "mike", "mike" };  int length1 = -9;  countRuns(d, length1); | Make sure the function returns -1 when it received a negative array length |

flip:

|  |  |
| --- | --- |
| Input | Reason |
| string cand1[5] = { "a", "A", "ab", "jill", "zjamu" };  int length1 = 5;  lookup(cand1, length1); | Make sure the function reverses the order of the elements of the array and return n |
| string cand1[5] = { "a", "A", "ab", "jill", "zjamu" };  int length1 = -5;  lookup(cand1, length1, "trump"); | Make sure the function returns -1 when it received a negative array length |

differ:

|  |  |
| --- | --- |
| Input | Reason |
| string cand1[6] = { "evan", "hillary", "mindy", "jill", "ajamu", "lmao" };  int length1 = 6;  string cand2[5] = { "evan", "hillarys", "mindy", "bill", "ajamu" };  int length2 = 5;  int result = differ(cand1, length1, cand2, length2); | Make sure the function returns the position of the first corresponding elements of a1 and a2 that are not equal |
| string cand1[6] = { "evan", "hillary", "mindy", "jill", "ajamu", "lmao" };  int length1 = -6;  string cand2[5] = { "evan", "hillarys", "mindy", "bill", "ajamu" };  int length2 = -5;  int result = differ(cand1, length1, cand2, length2); | Make sure the function returns -1 when it received a negative array length |
| string cand1[6] = { "evan", "hillary", "mindy", "jill", "ajamu", "lmao" };  int length1 = 6;  string cand2[5] = { "evan", "hillarys", "mindy", "jill", "ajamu" };  int length2 = 5;  int result = differ(cand1, length1, cand2, length2); | Make sure, if the arrays are equal up to the point where one or both runs out, the function returns whichever value of n1 and n2 is less than or equal to the other. |

subsequence:

|  |  |
| --- | --- |
| Input | Reason |
| string cand1[6] = { "evan", "hillary", "mindy", "jill", "ajamu", "lmao" };  int length1 = 6;  string cand2[2] = { "evan", "hillary" };  int length2 = 2;  int result = subsequence(cand1, length1, cand2, length2); | Make sure if all n2 elements of a2 appear in a1, consecutively and in the same order, the function returns the position in a1 where that subsequence begins |
| string cand1[6] = { "evan", "hillary", "mindy", "jill", "ajamu", "lmao" };  int length1 = -6;  string cand2[2] = { "evan", "hillary" };  int length2 = -2;  int result = subsequence(cand1, length1, cand2, length2); | Make sure the function returns -1 when it received a negative array length |
| string cand1[6] = { "evan", "hillary", "mindy", "jill", "evan", "hillary" };  int length1 = 6;  string cand2[2] = { "evan", "hillary" };  int length2 = 2;  int result = subsequence(cand1, length1, cand2, length2); | Make sure if the subsequence appears more than once in a1, the function returns the smallest such beginning position in the array |
| string cand1[6] = { "evan", "hillary", "mindy", "jill", "ajamu", "lmao" };  int length1 = 6;  string cand2[2] = { "evan", "hillary", "mindy", "jill", "ajamu", "lmao", "lol" };  int length2 = 7;  int result = subsequence(cand1, length1, cand2, length2); | Make sure the function returns -1 when n2 is larger than n1 |
| string cand1[6] = { "evan", "hillary", "mindy", "jill", "ajamu", "lmao" };  int length1 = 6;  string cand2[2] = { "evan", "bill" };  int length2 = 2;  int result = subsequence(cand1, length1, cand2, length2); | Make sure the function returns -1 when if a1 does not contain a2 as a contiguous subsequence |

lookupAny:

|  |  |
| --- | --- |
| Input | Reason |
| string cand[4] = { "gary", "hillary", "jill", "donald" };  int length1 = 4;  string cand2[6] = { "evan", "hillary1", "mindy", "jill", "ajamu", "lmao" };  int length2 = 6;  int result = lookupAny(cand, length1, cand2, length2); | Make sure the function returns the smallest position in a1 of an element that is equal to any of the elements in a2 |
| string cand[4] = { "gary", "hillary", "jill", "donald" };  int length1 = -4;  string cand2[6] = { "evan", "hillary1", "mindy", "jill", "ajamu", "lmao" };  int length2 = -6;  int result = lookupAny(cand, length1, cand2, length2); | Make sure the function returns -1 when it received a negative array length |
| string cand[4] = { "gary", "hillary", "jill", "donald" };  int length1 = 4;  string cand2[6] = { "evan", "hillary1", "mindy", "jill1", "ajamu", "lmao" };  int length2 = 6;  int result = lookupAny(cand, length1, cand2, length2); | Make sure the function returns -1 when no element of a1 is equal to any element of a2 |

separate:

|  |  |
| --- | --- |
| Input | Reason |
| string cand[6] = { "gary", "hillary", "jill", "donald" };  int length = 4;  int result = separate(cand, length, "hillary"); | Make sure the function rearranges the elements of the array so that all the elements whose value is < separator come before all the other elements, and all the elements whose value is > separator come after all the other elements. Then the function returns the position of the first element that, after the rearrangement, is not < separator, or n if there are no such elements. |
| string cand[6] = { "gary", "hillary", "jill", "donald" };  int length = -4;  int result = separate(cand, length, "hillary"); | Make sure the function returns -1 when it received a negative array length |
| string cand[6] = { "Gary", "Hillary", "Jill", "Donald" };  int length = 4;  int result = separate(cand, length, "hillary"); | Make sure the function returns n if no element in the array is < separator |