Zach North

603-885-768

*Project 4 Report*

To be honest I didn’t have many problems with this project. Everything seemed fairly straightforward, mainly because it was broken up into discrete chunks that were easy to handle one at a time. The *subsequence* function posed a bit of a problem because it called for a loop nested within a loop, but it still was not too tough to figure out. *Split* was very simple to implement using another declared array, but doing so without (to get the extra credit) was a bit of a challenge. However, using *rotateRight* in the solution solved everything perfectly and condensed the whole function to less than twenty lines of code.

Testing the ten different functions requires a variety of different input data.

* *setAll* requires an empty string array and a sample string to set all elements to. If it runs correctly, every element in the array up to integer *n* will be changed to the sample string.
* *Lookup, positionOfMax, rotateLeft, rotateRight,* and *flip* all require a string array with a variety of different values for each element. This way when you run each of the functions you can mentally check whether they give accurate results or not.
* *Differ* requires two separate and different string arrays that are able to be compared. Preferably they share some elements so the program can be tested both for error-checking functionality and that it avoids false positives.
* *Subsequence* requires two string arrays, one large and one small (only a couple elements), used to check that the smaller appears within the larger. The larger should contain some “triggers” to try to break the function (ex: 2 elements of the smaller in a row but not all 3.)
* *LookUpAny* can basically use 2 string arrays of any length; you can verify the results simply by looking at the two arrays side by side.
* *Split* requires a long string array and a sample string used to split the array. Preferably test data should include a variety of different sample strings to be tested – the empty string, a string identical to an array elements, etc.

The actual data I used is present in the main function of array.cpp. The sheer volume of possible strings for testing means that typing out every single test situation would take days, but I feel that I adequately tested the functions using my test input data. For all functions I made sure to test:

* Arrays with blank strings
* Arrays with all elements the same
* Arrays with multiple identical element values, both next to each other and separated
* All caps, all lowercase, mix of both
* Spaces