**Homework Two Project Report**

In this homework project, a key observation was the interrelation between pointers and arrays (pointers to char and char arrays) and how that is used in the implementation of C++ functions such as the standard library function strtok. The implementation of our own mystrtok further expanded on pointer and array manipulation as we take a string input and tokenize it by iterating through the memory and saving specific address points as starts and ends. In addition to separate helper file understanding, the decomposition of the strtok function into simpler parts with our homework1 functions (find\_first\_not\_in\_set and find\_first\_in\_set) shows the importance of scope of definition and implementations in order to have a compilation and run that behaves as expected.

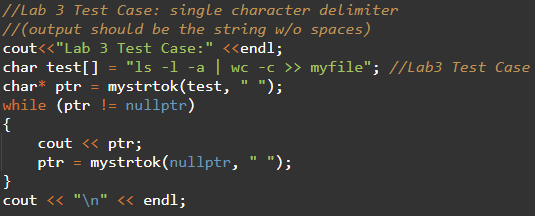
For a cstring to be valid, it must be a null-terminating string that can be represented as a char array. Null-terminating means that the character array/cstring ends with the null-character (‘\0’) in the memory. This lab improved my understanding of cstrings, null-terminating strings, and null-character by making us code for conditions in which the value of a variable is equal to the null-character and set a variable to the null-character.

The reason for having varying test inputs and predicted outputs is to cover a wider range of cases of the function being tested to see if it behaves as expected, especially when inside a loop.

Function: **mystrtok(char \*str, const char \*delim)**

**Lab03 Sample Test:**

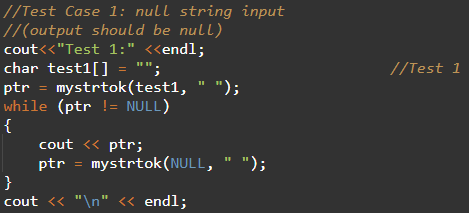
* **String= “ls -l -a | wc -c >> myfile” , Delimiter= “ ”** : tests a single character delimiter case
* Code:



* + Output:

**Test 1:**

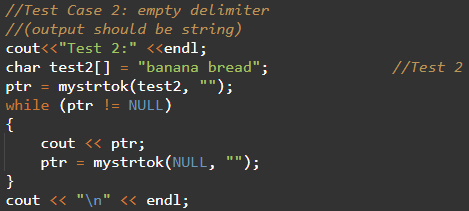
* **String= “” , Delimiter= “ ”** : tests a null string input case
* Code:



* + Output:

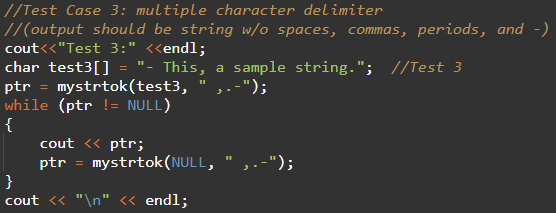
**Test 2:**

* **String= “banana bread” , Delimiter= “”** : tests an empty delimiter case
* Code:



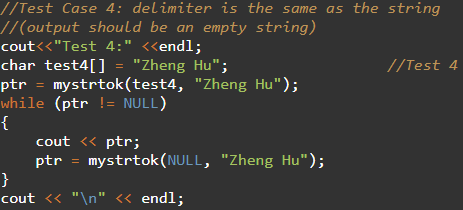
* + Output:

**Test 3:**

* **String= “- This, a sample string.” , Delimiter= “ ,.-”** : tests a multiple character delimiter case
* Code: 
  + Output:

**Test 4:**

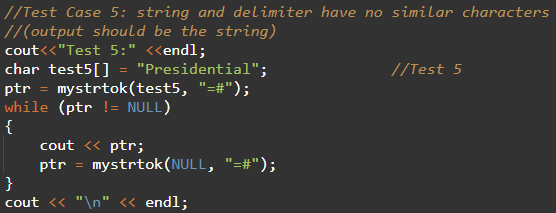
* **String= “Zheng Hu” , Delimiter= “Zheng Hu”** : tests a case where delimiter and string are equal
* Code:



* + Output:

**Test 5:**

* **String= “Presidential” , Delimiter= “=#”** : tests case of completely different string and delimiter
* Code:



* + Output: