CSCD 340
Lab 4

You answers will be in the form of a PDF/C Code.

1. One of the operations the shell performs is to parse a string into tokens. When you run your program with ./a.out argc will have the value of 1, and argv[0] will have the value of ./a.out. Understanding how to tokenize a string is an important concept. For this question you will emulate the parsing the shell does. You will need to complete several functions.

   Write a function that will parse a string into tokens (or words), similar to what the shell is required to do. The function is named makeargs.

   The prototype is given as: *int makeargs(char * s, char*** argv);*

   This function should accept a (c-type) string and a pointer to a two dimensional array representing the tokenized string. The function will return a 2D array of characters. Tokens are delimitated via whitespace '\t', '\n', ' '. The function will return an int representing the number of tokens in the string. Each row in the 2D array will be a separate token. The 2D array will contain one extra row that contains a null pointer. If a problem occurred during operation of the function, then return -1.

   For example, given the following C code

```
int main()
{
   char **argv, s[] = "ls -l file";

   int argc;

   argc = makeargs(s, &argv);

   printargs(argc, argv);

}// end main
```

   The results of *makeargs* would be:

   argc would be 3.
   argv[0] would be 'ls'
   argv[1] would be '-l'
   argv[2] would be file
   argv[3] would be '\0'

   You must not waste memory, and any memory you allocate you must clean up.

   I have provided as a starting point cscd340Lab4.c

   NOTE: The strings will be entered on the command line separated by a single space. You can presume the happy part of Stuland. HINT: You may need strtok and other string commands.

   DO NOT use **realloc,** use only **free** and **malloc**/**calloc**

FOR EXAMPLE

The user might enter: how now brown cow

And your program would report 4 strings and then print each string.

This will continue until the user types exit.

NOTES

- I have provided stubbed out code that you must use – You WILL NOT change my main or the file cscd340Lab4.c
- You must use strtok_r from string.h.
- For ease I have provided myUtils.h an myUtils.c – You may not change these files in any fashion and you must use them.
- You must use my Makefile – there should be no reason to modify it.
- Your output capture should have multiple outputs showing you truly tested your program. NOTE: the grader will be running **valgrind** on your program so make sure you clean up memory. These runs are stored in your PDF named lab4output.pdf

TO TURN IN

A zip

- Your PDF file
- All code necessary to compile and execute your program

You will submit a zip file named your last name first letter of your first name lab4.zip (Example steinerslab4.zip)