CSCD 240
Lab 10

In a normal C program you have int main(int argc, char **argv). We, however have been using int main(). It is important that we understand exactly how argc argv are determined. 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), like what the shell is required to do. The function is named makeargs.

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

This function should accept a (c-type) string and an int pointer that will represent the number of tokens in the string. Tokens are delimitated via whitespace. The function will return a 2D array of characters. Each row in the 2D array will be a separate token. 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;

   argv = makeargs(s, &argc);

   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

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

I have provided as a starting point cscd240Lab10.c, and some code in lab10.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 a few other string commands.

You will NOT use **realloc,**

You can use only **free, malloc/calloc, strlen, strcpy/strncpy and strtok**

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.

You must use strtok from string.h.

Your output capture should have at least outputs showing you truly tested your program. NOTE: the grader will be running **valgrind** on your program so make sure you clean up memory.

TO TURN IN

A zip of your Lab10 folder containing

- All your C/H files
- My Makefile
- Your output runs saved as cscd240Lab10out.txt
- Valgrind run named cscd240Lab10val.txt

You will submit a zip file named your last name first letter of your first name lab10.zip (Example steinerslab10.zip)