Lab 1 - Description

(String processing in C)

Lab Overview:

For this lab, we will be learning how to process complex strings from the console input. Typically, user I/O for operating systems CLI's takes the form of processing complex string input by the user into a shell/terminal or console. These commands are then parsed and sent to a sub-process that executes them using a wide variety of system calls or predesigned routines. In this lab, we will be going over how to take complex input from the console on a line-by-line basis and parsing that into understandable tokens that can be used by the system. We will be creating a helper structure that we can use to tokenize strings for current and future projects.

Core Tasks:

1. Use the provided skeleton file and implement provided header file functions.
2. Test your code with Valgrind for memory leaks.

Task Details:

1. Tokenize the input string. Utilize strtok_r(3): http://man7.org/linux/man-pages/man3/strtok.3.html
2. Implement string_parser.c (lab1_skeleton.c is your main program that is already implemented.) If string_parser.c is implemented correctly, your program will match the output file **exactly**.
3. Make sure no memory leak exists in your program as this program will be used for future projects.

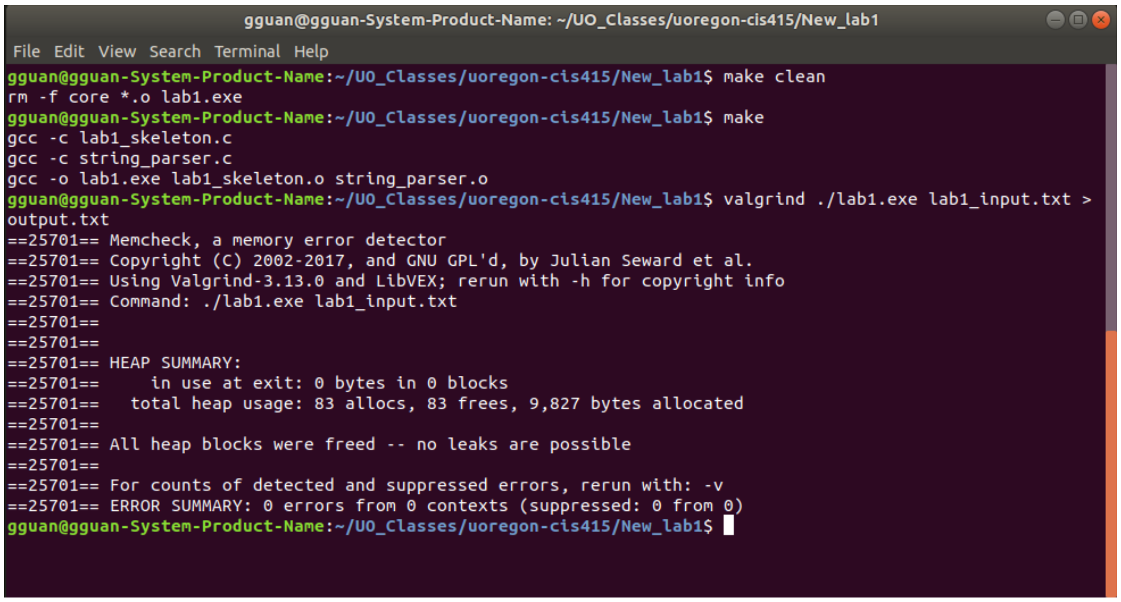
Provided Files:

1. lab1_skeleton.c
2. string_parser.c
3. string_parser.h
4. output.txt
5. lab1_input.txt
6. makefile

Submission Requirements:

In order to receive any credit for a lab, completion of the labs' core tasks must be demonstrated. You must submit the zip or tar file that should contain the following.

- string_parser.c
- output.txt
- a screenshot containing the following information


gguan@gguan-System-Product-Name: ~/UO_Classes/uoregon-cis415/New_lab1
File Edit View Search Terminal Help

```
gguan@gguan-System-Product-Name:~/UO_Classes/uoregon-cis415/New_lab1$ make clean
rm -f core *.o lab1.exe
gguan@gguan-System-Product-Name:~/UO_Classes/uoregon-cis415/New_lab1$ make
gcc -c lab1_skeleton.c
gcc -c string_parser.c
gcc -o lab1.exe lab1_skeleton.o string_parser.o
gguan@gguan-System-Product-Name:~/UO_Classes/uoregon-cis415/New_lab1$ valgrind ./lab1.exe lab1_input.txt >
output.txt
==25701== Memcheck, a memory error detector
==25701== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==25701== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
==25701== Command: ./lab1.exe lab1_input.txt
==25701==
==25701==
==25701== HEAP SUMMARY:
==25701==     in use at exit: 0 bytes in 0 blocks
==25701==   total heap usage: 83 allocs, 83 frees, 9,827 bytes allocated
==25701==
==25701== All heap blocks were freed -- no leaks are possible
==25701==
==25701== For counts of detected and suppressed errors, rerun with: -v
==25701== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
gguan@gguan-System-Product-Name:~/UO_Classes/uoregon-cis415/New_lab1$ 
```

The naming convention of the zip/tar file while uploading to canvas

- UoID_duckID_LAB/ProjectX (an example is given below)
  - UOID : alex
  - DuckID: 951505xxx
  - Submission for: Lab1
  - So the name of the zip/tar file should be:

    alex_951505xxx_Lab1.tar **or** alex_951505xxx_Lab1.zip