EE 4374 Operating Systems Design
Lab #1: Argument Tokenizer
Due Date: Lecture 2 Week 4 (before Midnight) [Feb. 11, 2016]

Objectives:

1) To review memory allocation and pointers in C.
2) To review how strings are represented in C.
3) To review how to organize source files.
4) To review how to use 'make' to build a software project.

Tasks:

1) Write an argument tokenizer function *argtok()*. The function prototype for argtok() must be in an accompanying header file. *firstinitiallastname_argtok.c/h* Example: mmcgarry_argtok.c/h

In general, tokenizers partition a character string into individual tokens (words) separated by reserved "delimiter characters". In this case, the delimiter character is the space character and tokens contain any characters other than space characters.

Your function "argtok" should have the following prototype:

char ** argtok(char *)

A call to argtok() should return a freshly allocated null-terminated array of pointers to freshly allocated strings containing each "token" from the input string. The input string can be of any length, and should not be modified. The vector returned by argtok() and the strings referenced by it, should all be allocated using malloc or calloc (so that it would be appropriate to free() them). Finally, these dynamically allocated blocks of memory should not be larger (or smaller) than required.

For example, a call to argtok("hello world") should return the address of an array containing:

{"hello", "world", 0}

Other than malloc/calloc/free, your implementation of argtok() should only call functions written by you. In particular, **calls to functions whose prototypes are in "string.h" such as "strtok()" are prohibited.**

2) Write file named *firstinitiallastname_lab1.c* containing a test program that includes the header file *firstinitiallastname_argtok.h*. The behavior of this program is described below.

Your test program should:

- print a prompt "$ "
- read a string typed at the keyboard.

break the string into tokens and print them out, one to a line.

3) Use a Makefile to build your program. You will rename your Makefile when you submit it.

4) Submit, via email, the deliverables described below. Use all lower case characters for your file names.

Deliverables:

1) Submit the C source code **with good comments**: firstinitiallastname_argtok.c/h and firstinitiallastname_lab1.c
2) Submit the Makefile **with good comments**: rename it to firstinitiallastname_Makefile
3) Submit your source code to the TA via email.

Scoring:

Lab grades will be determined by three criteria. The first criterion is whether your program compiles and runs with the correct result. The correct result must adhere to the **Tasks** section. The second criterion is whether your program uses the specified libraries and/or function prototypes as specified in the **Task** section. The last criterion is whether your source code is well documented. Your source code must include (at the top) your name, class section, due date, assigned date, and a small description of your program. Also, every function/method must be commented by specifying the purpose of the input values (if any) and their respective output values (e.g. void foo(int x) /* input: x > 0, output: x = x * 2*/).

Operation/Successful Demonstration	**60%**
Does the program compile? 30%	
Does the program run correctly? 30%	
Adherence to Interface Specification	**20%**
Does your program use the prototype functions specified? 10%	
Does your program use the libraries/APIs specified in the lab assignment? 10%	
Comments	**20%**
Is every method/function well-documented? 10%	

Is the source code well-documented at the top? 10%	
Lateness	**10%** per day (including weekends and holidays)