L70: Bonus Lab

Note: This work must be completed by the end of the lab on ____(See D2L)______.

Name: ___________________________________ ID: ___________________________________

Goals:

1. Be able to write professional C programs.

Tasks

- This lab is meant as a challenge and a bonus. If you complete it, passing <u>all</u> the supplied tests, this mark will replace your lowest lab mark, if this is to your advantage.
- Download the L70 support files from D2L. Do not change any files, except as outlined below.
- Implement bonus.c. See bonus.h for details.
- Fix every error caught by the tests.

Deliverables:

- Hand in a printout of your bonus.c file attached to this lab sheet.
- After handing in the printout, submit your <u>bonus.c</u> source code to D2L.
- You will not be able to make any changes once you have submitted the source code to D2L, so be sure to test your work thoroughly!

Rating	Correctness/Efficiency	Documentation	Structure/Complexity
***** perfect	- passes all tests	- well-documented, allowing another programmer to use all functions based on the header comments alone	- well-engineered, consisting of a modular collection of simple, single-purpose functions
	- code review reveals no faults	- responsibilities of all functions are described well, without giving implementation details	- constants are used whenever appropriate
	- efficient (given the requirements)	- all parameters, return values, and side effects are explained	- globals are not used, unless unavoidable
	- no redundant operations	- comments within all functions are helpful, without being distracting, making it easy to follow along	- all constants and variables are named appropriately
			- no layout abnormalities (eg, missing or improper indentation)
			- easily used and reused
			- no undesirable side effects, such as debug output
**** good	- passes nearly all tests	- occasionally, there are comments that are not complete, helpful, and/or true	- largely well-engineered, except for a few, minor issues
	- a code review reveals nearly no faults; faults that are found, are minor	- could be improved by slightly reworded, slightly more, or slightly fewer comments	- a few, minor issues with constants, variables, or layout
	- generally efficient, except in a few minor cases		- easily used and reused, except in a few, minor instances
	- at most a few redundancies		- generally no undesirable side effects
*** ok	- passes half the tests, or more, but the failure rate is too high for a 4-star rating	- in a number of cases, comments are cryptic, false, incomplete, misleading, missing, or redundant	- in need of reengineering due to a number of issues, none, or almost none of which, are major
	- a code review reveals a number of faults, but none, or almost none of them, are major		- on a number of occasions, there are issues with constants, variables, or layout
	- somewhat efficient, but there are a small number of major inefficiencies		- often easily used and reused, but there are a small number of major problems, none of which render the work unusable
	- potentially many redundancies		
** fail	- fails more than half the tests	- only little relevant documentation	- a number of major issues, requiring major changes
	- a code review reveals a high number of faults, including a number of major faults	- comments are often cryptic, false, incomplete, misleading, missing, or redundant	- not easily used and reused
	- not efficient, although slow progress is being made		
* fail	- fails nearly all the tests	- essentially no legitimate documentation	- only a few functions, many of which are responsible for too many things
	- code review reveals a proliferation of major faults		- essentially not usable or reusable
0 fail	- fails all tests and/or does not run	- no legitimate documentation and/or does not run	- no legitimate code and/or does not run

Correctness/Efficiency: _______ / 5 x 2 = _______ / 10
Documentation: _______ / 5
Structure/Complexity: _______ / 5

Total: _____ / 20 = _____ / 10