Grading guidelines

- Late Assignments will not be accepted except as covered under CSN policies.
- Look at the text of each individual assignment for specific guidelines on submission of files. Each assignment may have its own submission requirements. Assignments that do not adhere to the submission requirements will not be accepted.
- All programs must compile successfully without errors and warnings.
- Programs that do not compile will receive a grade of 0.
- All programs will be run through Stanford's Measure of Software Similarity (MOSS). MOSS is a software program that detects plagiarism. Any programming assignments that are deemed duplicates will receive 200-point deduction for the first offense. Subsequent cheating cases are handled through the academic misconduct policy.
- Any program that has runtime errors will receive a 50% deduction. Partial grade will be assigned for logical errors.

Coding style & documentation

All program files must have the correct file name and type, and have the comment block section as **specified at the end of this document.** The header comment section must be placed at the top of the main program file.

Variables:

i. Use descriptive names for variables using naming standards discussed in class. Always, start a variable name with a lower case letter. Global variables are not permitted; global constants are permitted.
ii. Declare one variable per line.
iii. Document the purpose of every variable.

Basic blocks (code enclosed in braces):

i. Blocks should use braces using methods demonstrated in class: Must be on separate lines or the beginning brace can be on the statement before the block begins.
ii. Statements in the block should be indented consistent with logical nesting: 4 spaces per indent level. **Do not use tabs.**

Statements:

i. No more than one statement may be written on a single line.
ii. The following may not be used: *continue*, *goto*, and *break*. The switch structure is an exception where break must be used to break out of a block.
iii. Lines of code should not extend beyond column 80.
iv. Diagnostic/debug print statements should be disabled/deleted in the final submission. (Submit a “shipping” version of your program.)

Functions:

i. Use descriptive names for functions using naming standards discussed in class. For example, Function names must start with a lower case letter. Class names start with a capital letter.
ii. A brief description of each function, when applicable, should be placed in comments directly above the function header at the function definition, not prototype. Document the purpose of all arguments and return values.
iii. Function bodies should not be of extended length when easily separated into multiple functions. (Functions should do one thing.)
iv. Functions should contain at most one **return** statement.
v. All functions must have the following required documentation immediately above the function class definition:
/* function_identifier: brief description of what the function does.
* parameters: what to pass into the function
* return value: what the function returns, if any */
All programs must have reasonable comments throughout the program that properly describe the process

Syntax & general knowledge

i. All variables, functions, arrays & structs must have meaningful identifiers.
ii. All programs must avoid the use of global variables.
iii. All programs will refrain from using hard-coded values in the program. You must use named constants instead of hard-coded values.

Program solution

All programs must meet the given requirements in the text of each assignment. Thoroughly test your code. For example, if your code requires that a given input value to be a positive number, you must test for input of negative values. In most cases, I provide "sample runs" of your code. Be advised that you must test your code beyond the sample cases provided in the text of your assignment.

Documentation requirements:

In order to properly document your code, all your programs must have the following minimal information. You may add additional documentation as you see fit.

Name:

Class name:

Date Assigned:

Date Due:

A brief explanation of the program.

Input variable list and a brief explanation of each main variable such as:

FurnType – The type of furniture

FurnSize – The size of furniture

Output variable list and a brief explanation of each variable such as:

Cost – The furniture cost

Time taken to complete this program:

Note: List the main variables that you are using in your program. If you are writing a program to calculate income tax, then variables such as income, tax rate, filing status, etc. become the main variables in a program.

Document the critical steps of the program such as calculation of the average, the loop that displays the output (we're not using a loop for this assignment), or the place where the furniture cost is calculated. For instance, I may have the following line of documentation in my code:

/* The user is prompted for furniture type and size */

Or,

//Calculate average grade

In case you are writing your pseudocode in your CPP file, you can write something like:

/*****************

1. start

2. set x = 10

3. set y = 20

4. etc.

**********/**