Assignment 1		**Student:** Austin Major	
Max Value	**Earned Value**	**Check Marks**	**Instructor Comments**
Submission (10)			
5	5	Source/header filenames correct and named using UpperCamelCase	
5	3	Comment lines identify filename, author, student ID, assignment #	Filename comment incorrect in Ticket.h and Ticket.cpp. Fix this for the next assignment.
Programming Style (36)			
6	4	Exhibits continuity of style throughout project in accordance with the Linux Kernel Coding Style and any additional requirements included in this course's style guide • Use of open/close parenthesis (),no parentheses between function name and opening parenthesis and a single space between operand and opening parenthesis of expression • Use of open/close curly braces {}, opening curly brace for functions always on newline, everywhere else always at end of line with a single space preceding curly brace • Use of open/close brackets [] • Use of indentation, one level of indentation after each opening curly brace until closing curly brace • Use of blanks lines to separate functions, sections of code • Spacing around operands (*if-else, while, do-while, for, switch)* • Spacing around binary and unary operators	Make sure style is followed. Always precede the opening { brace at end of line with single space. Always include a single space before and after ALL binary operators. Use a single space to separate a statement operand from the opening parenthesis. Functions on the other hand do not have a space separating function name from opening parenthesis. Fix all of the above for the next assignment.
6	6	Identifier names • Classes are named using UpperCamelCase • Function and variable names are meaningful and are named using lowerCamelCase, short names (one char in len) are allowed for simple counter variables	Good
6	6	Scope • Variables declared with minimal scope (declared in the innermost block required and no global declarations)	Good
6	6	Flow control • Exhibits good use of flow control and loop statements (*if-else, while, do-while, for, switch*)	Good
6	6	Classes and functions • Classes demonstrate OOP principle of data encapsulation • Functions perform specific tasks (black-box style)	Good
6	5	Commentary • Exhibits good use of commentary throughout header and source	Always document a function by placing comments prior to the function definition.

		files, comments are meaningful	Always have a single blank line preceding a function to separate it from the other functions (so this means the blank line would be above the comment for the function). Fix all of the above for the next assignment.
Accuracy (30)			
10	10	Source files compile without syntax errors (use –Wall flag) and include sufficient logic to produce expected outcomes	
10	10	Program runs without causing a run-time error using the "happy path", only valid values (conversion errors are tolerated for this assignment) and exhibits sufficient logic to produce expected outcomes	
10	10	Program's input and output are user-friendly, easy to understand and use and exist in the same file as the main() function. A specialized class should not perform input and output.	
Test Cases for Input (9)			
3	3	Input prompts for ticket number allowing alphanumeric, but no embedded spaces	
3	3	Input prompts for gross and tare weights as integers	
3	3	Input prompts for moisture level and foreign material as percentages, float or double	
Test Cases for Output (15)			
3	3	Output shows alphanumeric ticket number	
3	3	Output shows gross and tare weights as integers	
3	3	Output shows gross and net bushels as float or double	
3	3	Output shows moisture level and foreign material in bushels as float or double	
3	0	Output shows sample's percentage for moisture level and foreign material	Cannot simply embed a constant value into the string for output for these values. Values must reflect what the user entered.
100	92.00	**Total**	