Assignment 3		**Student:** Austin Major	
Max Value	**Earned Value**	**Check Marks**	**Instructor Comments**
Submission (10)			
5	5	Source/header filenames correct and named using UpperCamelCase	
5	5	Comment lines identify filename, author, student ID, assignment #	
Programming Style (30)			
5	5	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 (multiple blank lines condensed to a single line) • Spacing around operands (*if-else, while, do-while, for, switch)* • Spacing around binary and unary operators	Use the same char(s) for indenting throughout project. The Ticket.h file uses spaces while the source files use a tab char.
5	5	Identifier names • Function and variable names are meaningful and are named using lowerCamelCase, short names (one char in len) are allowed for simple counter variables • Classes are named using UpperCamelCase • A constant variable is used instead of an explicit value when the value is subject to change or used repeatedly	
5	5	Scope • Variables declared with minimal scope (declared in the innermost block required and no global declarations)	
5	5	Flow control • Exhibits good use of flow control and loop statements (*if-else, while, do-while, for, switch*)	
5	5	Classes and functions • Classes demonstrate OOP principle of data encapsulation • Functions perform specific tasks (black-box style)	Since the friend non-member overloaded operator << was declared in the Ticket class and is intended to be a feature of the class, include its implementation in the Ticket source file, not the

		• Constants for a class are limited in scope to the class implementation itself using unnamed namespace • *Ticket* class implements member function *toString()* to return receipt of ticket • *Ticket* class implements overloaded equality operator == to compare two Ticket objects • *Ticket* class implements overloaded friend insertion operator << to return receipt using *toString()* member function	WheatHarvest source file.
5	5	Commentary • Exhibits good use of commentary throughout header and source files, comments are meaningful • Functions are commented • Logical groups of statements are commented	
Accuracy (40)			
20	20	Source files compile without syntax errors (use –Wall flag) and include sufficient logic to produce expected outcomes • The main() includes a check for duplicate tickets by comparing two *Ticket* objects for equality (does not compare ticket numbers directly for equality as that is done within the overloaded equality operator function in the class implementation) • The main() outputs the ticket receipt using the Ticket class overloaded insertion operator <<	
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 (10)			
5	5	Input prompts for ticket information using same requirements as assignment 2	
5	5	Input checks for duplicate ticket number and displays a message when a duplicate is found	
Test Cases for Output (10)			
5	5	Output shows ticket information (receipt) for each ticket using same requirements as assignment 2	
5	5	Output shows summary of total gross and net bushels for all tickets using same requirements as assignment 2	
100	100.00	**Total**	Well done!