WICHITA STATE UNIVERSITY

CS 311: Object Oriented Programming
Spring 2018 – Assignment 3

For this assignment, we will improve upon the program implemented in assignment 2 by introducing a change to prevent duplicate ticket numbers. In addition, some refactoring of the code will demonstrate the use of namespaces and overloaded operators for the *Ticket* class as well as a function to return a string representing a ticket receipt. In addition, any issues raised in the feedback you received for assignment 2 must addressed (fixed) as part of this assignment.

Use the following criteria for writing your program:

- ✓ Modify the program named *WheatHarvest* (source file named WheatHarvest.cpp) to:
  - Use an overloaded equality operator to compare two Ticket objects for equality. Compare the ticket entered to the list of tickets entered thus far (hint: use a loop to compare ticket to each ticket in list, but be sure you are comparing two ticket objects) to prevent duplicate tickets (tickets with the same ticket number) and output an error when a duplicate occurs
  - Use an overloaded insertion operator during the output of a ticket receipt

- ✓ Modify class named *Ticket* (header file named Ticket.h, source file named Ticket.cpp) to:
  - Include a member function named *toString()* to format and return the ticket receipt as a string (format of string should look like the output of a ticket seen in the previous two assignments)
  - Include a member function to overload the equality operator that returns *true* if the ticket number is the same as the number associated to the ticket provided as a parameter, otherwise returns *false*
  - Include a friend function to overload the insertion operator and output the receipt associated to the ticket provided as a parameter (hint: use the member function *toString()* to obtain the ticket's receipt as a string)
  - Use unnamed namespace(s) for all constants used by the implementation of the *Ticket* class, including those declared as global/static as well as those defined in the private section of the *Ticket* class definition (remove these from the private section)

To test the changes, enter a duplicate ticket number and ensure an error message occurs. The remaining changes are a result of refactoring, so the user should not see any noticeable differences.

Example Input:

Ticket number 1 (alphanumeric): 101300A
Gross weight (lbs): 33180
Tare weight (lbs): 10780
Moisture level (%): 14.0
Foreign material (%): 0.75

Ticket number 2 (alphanumeric): 101400A
Gross weight (lbs): 24150
Tare weight (lbs): 10780
Moisture level (%): 11.5
Foreign material (%): 1.25

Ticket number 3 (alphanumeric): 101300A
Gross weight (lbs): 31200
Tare weight (lbs): 10780
Moisture level (%): 13.25
Foreign material (%): 2.0

Error: Duplicate ticket encountered, ticket ignored!

Ticket number 3 (alphanumeric): <nothing entered, user just presses Enter>

Example Output (based on input):

Ticket 101300A:
33180 Gross Weight
10780 Tare Weight
22400 Net Weight

373.33 Gross Bushels
7.47 Moisture Level (14%)
2.80 Foreign Material (0.75%)
363.07 Net Bushels

Ticket 101400A:
24150 Gross Weight
10780 Tare Weight
13370 Net Weight

222.83 Gross Bushels
0.00 Moisture Level (11.5%)
2.79 Foreign Material (1.25%)
220.05 Net Bushels

Wheat Harvest Summary Totals
596.17 Gross Bushels
583.11 Net Bushels

Requirements and Submission:

- Due date: Saturday, February 10, 11:59pm
- Include the following header information (comment lines) at the beginning of each source/header file (before the #include directives) submitted. Notice each comment line begins with two slashes. C++ recognizes these as the start of single line comment. If you prefer, you may use block commenting.

// File Name: WheatHarvest.cpp	Such as: // File Name: WheatHarvest.cpp
// Author: Firstname Lastname	// Author: Joe Shobe
// Student ID: ********	// Student ID: xxxxxxxx
// Assignment Number: ##	// Assignment Number: 3

- Grades will be based on:
  - ✓ User-friendly interactive input, output, and error messages (when applicable)
  - ✓ Programming style: Proper use of control statements, standard C++ techniques and naming conventions, and the conciseness and readability of the code
  - ✓ Accuracy: Does the program implement required changes, calculate and output correct data
  - ✓ Appropriate and descriptive comments
- Use of solutions found on the web, in the books, or from other students is <u>not</u> permitted. It must be your own work. Under no circumstances are you to share your work with another student.
- Review the section *Submitting Assignments* in the Linux Server PDF for more information on how to turn in your assignment.
- No submissions will be accepted by Blackboard, e-mail, CD, DVD, or other independent storage or electronic media, unless approved by the instructor.