WICHITA STATE UNIVERSITY

CS 311: Object Oriented Programming
Spring 2018 – Assignment 4

For this assignment, modify the program for assignment 3 to implement a new *Tickets* class that uses the heap for storing its array of objects. Also, implement a new *Grain* class to represent the sample of grain (ie. wheat) taken and new *Input* / *Output* classes containing static member functions do perform all input and output to the console. In addition, any issues raised in the feedback you received for assignment 3 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:
  - Rename *WheatHarvest* to *Harvest* (source file named Harvest.cpp)
  - Move all input processing to the new class *Input* (see below) and all output processing to the new class *Output* (see below). As a result, the main() will be simplified and should now have only four statements (with proper commentary of course)
    1. Declare a variable of the newly created class type *Tickets*
    2. Invoke function *inputTickets()* within *Input* class to initialize the list of tickets (variable of type *Tickets*) based on user input
    3. Invoke function *outputTickets()* within *Output* class to output the list of tickets
    4. Invoke function *outputSummary()* within *Output* class to output harvest summary
       (Hint: move logic to accumulate totals for list of tickets to this function)

- ✓ Implement class named *Tickets* (use header file named Tickets.h provided, so **DO NOT WRITE YOUR OWN**, source file named Tickets.cpp) to:
  - Initialize each member variable to 0 for the default constructor (hint: when a class member variable is of a class type, how does it get initialized?)
  - Implement a copy constructor to correctly copy a ticket
  - Implement a destructor to correctly destroy a ticket
  - Implement a member function to add a ticket to the list (array) of tickets
  - Implement a member function to return the size (number of tickets) of the list (array) of tickets
  - Implement an overloaded member assignment operator = to correctly assign a ticket
  - Implement an overloaded member array operator [] that returns a ticket based on the index value provided in the array subscript

- ✓ Implement class named *Grain* (use header file named Grain.h provided, so **DO NOT WRITE YOUR OWN**, source file named Grain.cpp) to:
  - Initialize each member variable to 0 for the default constructor
  - Initialize each member variable to value of parameters provided to the overloaded constructor
  - Implement function to return value of a file-scoped constant named AVERAGE_TEST_WEIGHT whose value is 60.0
  - Implement function to return value of a file-scoped constant named IDEAL_MOISTURE_LEVEL whose value is 12.0
  - Implement accessor functions to return values for each member variable

- ✓ Modify existing class named *Ticket* (header file named Ticket.h, source file named Ticket.cpp) to:
  - Include a new member variable of type time_t to store the timestamp taken when the ticket is created (its value is set only by constructors, no mutator function. Also, refer to the system time() function for obtaining system time)
  - Replace member variables for moisture level and foreign material with a single *Grain* member variable representing the grain sample taken

- Modify overloaded constructor to allow caller to pass a *Grain* reference parameter in lieu of moisture level and foreign material parameters (Hint: consider passing an anonymous object of type *Grain* as an argument to the overloaded constructor)
- Ensure each member variable is initialized to 0 for the default constructor
- Remove accessor functions to return values for moisture level and foreign material
- Implement accessor functions to return values for timestamp and *Grain* member variables
- Modify use of moisture level and foreign material to use accessor functions to return values for *Grain* member variables instead
- Modify *toString()* member function to include output of timestamp using the format MM/DD/YYYY HH:MM:SS

✓ Implement class named *Input* (use header file named Input.h provided, so **DO NOT WRITE YOUR OWN**, source file named Input.cpp) to:

- Prevent an object of type *Input* from being declared (hint: default constructor is private)
- Implement static member function to prompt for ticket information, create ticket, check for duplicate ticket, add ticket to list (array) of tickets if not duplicate

✓ Implement class named *Output* (useheader file named Output.h provided, so **DO NOT WRITE YOUR OWN**, source file named Output.cpp) to:

- Prevent an object of type *Output* from being declared (hint: default constructor is private)
- Implement static member function to process list (array) of tickets, outputting each ticket's receipt using overloaded insertion operator
- Implement static member function to process list (array) of tickets, accumulating totals, outputting harvest summary after all tickets are processeds

To test the changes, use sample input below and validate output. Most of the changes are a result of refactoring, so the user should not see any noticeable differences except the addition of a timestamp on the receipt for each ticket.

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 – 02/13/2017 16:05:00:
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 – 02/13/2017 16:05:30:
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

Harvest Summary Totals
596.17 Gross Bushels
583.11 Net Bushels

Requirements and Submission:

- Due date: March 1, 11:59 (old due date: Saturday, February 24, 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: Harvest.cpp
// Author: Firstname Lastname
// Student ID: ********
// Assignment Number: ##
```

Such as:

```
// File Name: Harvest.cpp
// Author: Joe Shobe
// Student ID: xxxxxxxx
// Assignment Number: 4
```

- 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.