CS/CE 1337 – PROJECT 4 – Ticket to Knowhere

Pseudocode Due: 3/30 at 11:59 PM
Project Due: 4/14 at 11:59 PM

Submission and Grading: All programs and pseudocode are to be submitted in eLearning. The pseudocode should be submitted as a Word or PDF document. Pseudocode is not accepted late. Please submit the all .cpp files and .h files together in one submission. **Do not zip the files. Attach each one separately in an eLearning submission.** Projects submitted after the due date are subject to the late penalties described in the syllabus. Programs must compile and run in Code::Blocks. Each submitted program will be graded with the rubric provided in eLearning as well as a set of test cases. These test cases will be posted in eLearning after the due date. Each student is responsible for developing sample test cases to ensure the program works as expected.

Problem: With the Guardians of the Galaxy making Knowhere their base of operations, that sector of space has become rather popular. This popularity brings with it extra traffic and is putting a strain on the intergalactic docking station. Ships are now required to pay for time at the docking station. Those ships that stay longer than their time allotment will receive an automated ticket. You have been hired to write a program that will help generate tickets for parking violations at Knowhere.

Functionality: This program will utilize a linked list as a queue. The ships will dock at the station if there is room and if not the ships will form a queue. Before docking, the ship will pay for time at the docking station. When the ship undocks, the time will be checked to see if the ship stayed longer than the time purchased. If so, a ticket will be generated. When a ship undocks, if there are ships in the queue, the first ship in the queue will dock at the docking station.

Classes:

- Linked List
  - Head (Node pointer)
- Node
  - Name (character pointer)
  - Ship ID (character pointer)
  - Credit (in minutes) (integer)
  - Node pointer (to point to next ship in the queue)
- Meter
  - Hour In (integer)
  - Minutes In (integer)
  - Node pointer (to point to docked ship)
- Ticket
  - Ship Info (Node)
  - Meter Info (Meter)
  - Hour Out (integer)
  - Minutes Out (integer)

Details:

- This project will utilize multiple classes (see above).

- Each class must have a header file and a source file.
- All data in the class should be private with proper mutators and accessors for public interface.
- Each class should have a default and overloaded constructor.
- The Ticket class should have an overloaded output operator.
- The Node class should have an overloaded input operator.
- You may add other functions into the class as necessary.
    - Functions added to the class should pertain to that object only.
- The docking station should be an array of Meter pointers
- There are 10 docks.
- You may assume that no ship will stay more than 24 hours over their credited time.

Input: All input will come from a file named `Knowhere.dat`. The file will contain multiple lines of information to describe what is happening at Knowhere. Each line will be one of the formats as described below. You do not need to validate any data. There will be spaces between each item on the line. You may assume that the entries in the file are in chronological order.

- *Ship entering Knowhere*
  `enter <time in> <ship name> <ship ID> <paid time>`
  Example
  `enter 14:15 Milano GG-08-2014 20`
- *Ship leaving Knowhere*
  `exit <ship ID> <time out>`
  Example
  `exit GG-08-2014 14:30`

For ships entering Knowhere, the time in is only valid if there is a spot available at the docking station. If the docking station has room, the ship will dock at the time specified. If there is no room at the docking station, put the ship in a queue. When a ship leaves the docking station, the first ship in the queue will dock 15 minutes after the ship leaves the docking station.

When a ship enters Knowhere, use the overloaded input operator of the node class to fill in the node (created dynamically).

Output: All output will be written to a file named `Tickets.txt`. This file will contain all of the tickets generated by ships that stayed at the docking station longer than the amount of time paid. When a ticket is generated, you must calculate the fine (I suggest a function in the parking ticket class to calculate this). The fine is 2,000 units per extra minute. The ticket should be of the following format:

```
Ship ID: <Ship ID>
Name: <Ship Name>
Time In: <Hour In>:<Minutes In>
Time Out: <Hour Out>:<Minutes Out>
Credited Minutes: <Credit>
Extra Minutes: <Minutes Over>
Fine: <fine> units
<blank line>
```

After the ticket has been created, use the overloaded output operator to write the ticket to the file.