PIC 10B Lectures 1 and 2 Spring 2015
Homework Assignment #8

Due Friday, June 5, 2015 by 6:00pm.

Objectives:

1. To use priority queues to help run a discrete event simulation.
2. To use a map to keep track of customer data.

Introduction:

This problem is taken from Exercise P13.14 (p.543). in *Big C++* by Horstmann.

A **priority queue** is a collection of items organized in such a way that permits fast access and removal of the largest element.

One application of priority queues is storing events in a **discrete event simulation**. We will be modeling customer arrivals and departures at a small hot dog stand that has only three seats. Departures and Arrivals will be modeled as objects with common base class Event. The base class Event and its derived class Departure are provided for you. You will be implementing Event subclass Arrival whose header file is provided.

The Arrival class will be overriding its inherited method act, which processes its event. The arrival constructor will initialize its members in a manner similar to Departure.

The act function of Arrival

The act function of Arrival will check to see if there is an available seat for its customer. If so, the customer "sits" by being added to the Simulation's seat vector Method act then displays the arrival time, followed by the message " Customer is seated." To format the time appropriately (2 decimal places), use the command

```
cout << fixed << setprecision(2);
```

before printing to the screen. Generate a random number between 1 and 4 to represent the number of hot dogs ordered by that customer. Record the time seated as the key and the number of hot dogs ordered as the value in the `map<double, int> hot_dog_orders`. Then create and schedule a Departure event by pushing it onto the Simulation's `priority_queue` called event_queue. The Departure event should be initialized with some random time that is later than the customer's arrival time. (Use the `get_time_later()` inherited method from Event).

If there is no seat, act will just create a new Departure event with some random time that is later than the customer's arrival time. The Departure should be initialized with message "Customer unable to find a seat, leaves."

Implementing the Simulation class

In addition, you will be defining the Simulation class whose header file is also provided. The Simulation class which models customers arriving and departing from the hot dog stand at various times.

Constructor

When a Simulation object is created, its constructor will set the random number generator seed (See PIC 10A Lecture 9: Random Numbers) and generate a random number of customer arrivals (ranging from 0 to 25) modeled by newly created Arrival objects, which you will push onto the priority queue one at a time. To create an Arrival object, construct a new Customer object on the heap and generate a random time modeled as a double. The formula for generating a random time in the interval [.01, 1.01] is

```
double time = 1.0*rand()/RAND_MAX + .01;
```

You will need to include `<cstdlib>` and `<ctime>` in the Simulation.cpp file. This will model a random number of arrivals of customers coming to your hot dog stand at various times looking for an available seat. The arrivals are all stored in the Simulation's event_queue which will retrieve every event (Arrival or Departure) in chronological order.

`run()`: The "event loop"

The Simulation member function `run()` will process all the events in the event_queue until there are no more by using an "event loop". The code on page 528 in your textbook will be helpful. (However, in your "event loop" be sure to pop the event off the event_queue before processing the next event.) In the event loop, current_time is the member of the Simulation representing the "system clock". The event_queue method top() will always return the next event chronologically, no matter in what order the events were inserted into the priority queue. Finally print a report to the screen showing when hot dogs were ordered and how many as in the screen shot below. You should also state the total number of dogs ordered as the last line.

For the simulation, we will model two possible scenarios:
Case 1: There is a seat available at the time of a customer arrival
Case 2: There is not an available seat upon arrival.

We decided not to take into account every possible scenario a customer could have for Case 2. To keep things simple, we decided that a customer who cannot find a seat upon arrival, will wait a random period of time and then just leave. There could be several reasons why a customer would not take a seat later should one become available. Perhaps the user doesn't want the available seat for some reason. For example:

- Somebody is talking loudly on their cell phone at the adjacent seat.
- The area near the available seat has not been cleaned.
- The customer realized he/she would not have enough time to eat.

Note that our scenario can also model impatient customer who leaves immediately since it is possible for get_time_later() to return the arrival time.

In short, we chose not to model the case when a waiting customer eventually takes an available seat. (I tried to keep the project short.) However, modeling this case would make the simulation a bit more realistic and interesting.

Directions

1. Create the project called “Hw8” in a solution called “Homework” using Microsoft Visual Studio 2012. All header (.h) and source (.cpp) files inside the project should contain the following header:

```
/*
   <Your Name>                      PIC 10B Intermediate Programming
   ID: <Your Student ID>            Spring 2015
   Email: <Your Email Address>      Assignment #8
   Section: <Your Section # eg 1A>
   Honesty Pledge:

      I, <Your Name Here>, pledge that this is my own independent work,
   which conforms to the guidelines of academic honesty as described in
   the course syllabus.

   List of known bugs:  <Known bugs, if any>
*/
```

2. Include the given files Customer.h, Customer.cpp, Event.h, Event.cpp, Arrival.h, Simulation.h, SimulationApp.cpp posted on CCLE’s Homework #8 Files in your Hw8 project.

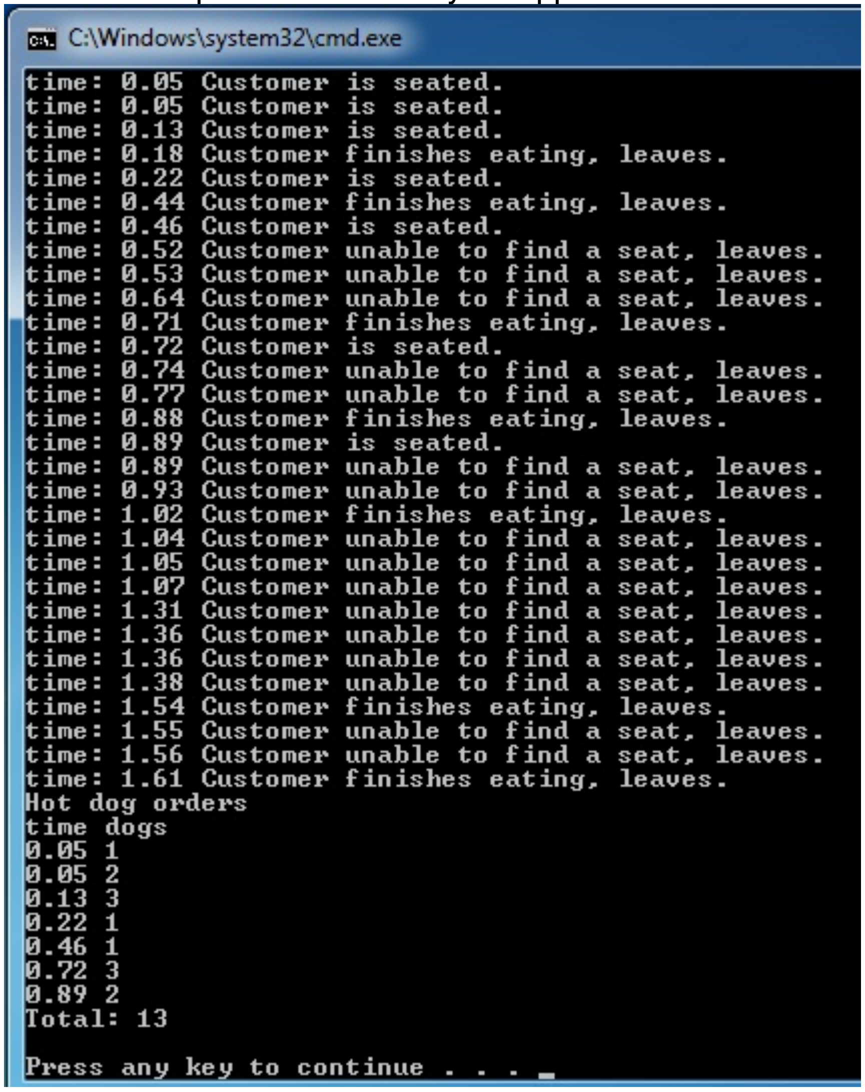
3. Implement the Arrival class according to the specifications above and include i the implementation file Arrival.cpp in your project.

4. Implement the Simulation class according to the specifications above and include the implementation file Simulation.cpp in your project.

5. When you have completed your project, be sure to
   - make sure your program compiles in Visual Studio 2012.
   - run your program to make sure it works correctly
   - upload your source code files
     - `Arrival.cpp`
     - `Simulation.cpp`

     using the CCLE website. No hardcopies will be collected.
   - visually verify that your source code was submitted correctly by clicking on the links to those files on the CCLE page after submission.

Here is a sample screenshot of your application:



```
time: 0.05 Customer is seated.
time: 0.05 Customer is seated.
time: 0.13 Customer is seated.
time: 0.18 Customer finishes eating, leaves.
time: 0.22 Customer is seated.
time: 0.44 Customer finishes eating, leaves.
time: 0.46 Customer is seated.
time: 0.52 Customer unable to find a seat, leaves.
time: 0.53 Customer unable to find a seat, leaves.
time: 0.64 Customer unable to find a seat, leaves.
time: 0.71 Customer finishes eating, leaves.
time: 0.72 Customer is seated.
time: 0.74 Customer unable to find a seat, leaves.
time: 0.77 Customer unable to find a seat, leaves.
time: 0.88 Customer finishes eating, leaves.
time: 0.89 Customer is seated.
time: 0.89 Customer unable to find a seat, leaves.
time: 0.93 Customer unable to find a seat, leaves.
time: 1.02 Customer finishes eating, leaves.
time: 1.04 Customer unable to find a seat, leaves.
time: 1.05 Customer unable to find a seat, leaves.
time: 1.07 Customer unable to find a seat, leaves.
time: 1.31 Customer unable to find a seat, leaves.
time: 1.36 Customer unable to find a seat, leaves.
time: 1.36 Customer unable to find a seat, leaves.
time: 1.38 Customer unable to find a seat, leaves.
time: 1.54 Customer finishes eating, leaves.
time: 1.55 Customer unable to find a seat, leaves.
time: 1.56 Customer unable to find a seat, leaves.
time: 1.61 Customer finishes eating, leaves.
Hot dog orders
time dogs
0.05 1
0.05 2
0.13 3
0.22 1
0.46 1
0.72 3
0.89 2
Total: 13

Press any key to continue . . .
```

Grade Breakdown:

Criteria	Description	Points
Header	Starts every .h and .cpp file	1
Comments	Program well-commented.	1
Arrival.cpp	`Arrival constructor`	3
	`act()`	10
Simulation.cpp	`Simulation constructor`	8
	`run()`	7
Total		30

A penalty of 5 points will be assessed if your code does not compile using Microsoft Visual Studio 2012.