CSCI 2170 Spring 2011 OLA7
(Electronic program submission due: midnight, Wednesday April 27th,
Hard copy of the program and turn in of the evaluation sheet are not required for this assignment)

Implement the event-driven simulation of the activities in a bank. A queue of arrival events will represent the line of customers in the bank. Maintain the arrival events and departure events in an ADT event list, sorted by the time of the event. Your program must compute the **average waiting time** and the **maximum waiting time** of the customers, and compute the **average length of the waiting queue** and the **maximum length of the waiting queue**. This requires your program to keep track of statistics including: the number of customers served, the cumulative waiting time, the maximum waiting time, the cumulative waiting queue length, and the maximum length of the waiting queue, etc..

Two additional requirements:

- When the arrival time of the arrival event is the same as the departure time of a departure event in the event list, the arrival event should be put in front of the departure event in the event list;
- In the situation when customer B comes in the bank at the same time when customer A is leaving, the length of the waiting queue = size of the current queue - 1.

Data file:
The data file contains arrival information. Each line of the data fie contains the arrival time and required transaction time of a customer. The arrival times are ordered by increasing time. Copy the data file into your directory with:

```
ranger% cp   ~cen/data/bank.dat   bank.dat
```

Program output:
Your program should display a trace of the events executed, as well as the final statistics.
Here is an example program output:

Simulation begins:

```
Processing an arrival event at time:      1
Processing an arrival event at time:      4
Processing a departure event at time:     7
Processing an arrival event at time:      11
Processing an arrival event at time:      12
Processing a departure event at time:     13
….                                        ..
```

Final Statistics:

```
Total number of people served:            15
Average time a customer spent waiting:    5.6 minutes
Maximum time a customer spent waiting:    17 minutes

Total time during simulation:             210
Maximum length of the waiting line:       6
```

Pointer-based implementation of ADT queue, and the pointer-based implementation of ADT EventList described in book and discussed in class should be used in this assignment.

Your program should include the following files:

- **type.h** -- defines the arrival/departure record structure type
- **type.cpp** – implements overloaded operator(s) defined for the arrival/departure record
- **queue.h** -- define pointer based implementation for **Queue ADT**
- **queue.cpp**
- **eventList.h** -- define pointer based implementation for **Sorted Event List ADT**
- **eventList.cpp**
- **ola7.cc**

To turn in the program, follow these steps:

- **Soft copy:**
  - login the ranger system with www.cs.mtsu.edu/nx,
  - login to PeerSpace through the web browser provided by the ranger system, click on tools|Assignments to submit your softcopy.

  Create a script file by following the steps below:
  First, navigate to the directory where your program source file is located, then follow the steps below:

```
ranger% script log7
ranger% pr –n –e4 type.h
ranger% pr –n –e4 type.cpp
ranger% pr –n –e4 queue.h
ranger% pr –n –e4 queue.cpp
ranger% pr –n –e4 eventList.h
ranger% pr –n –e4 eventList.cpp
ranger% pr –n –e4 ola7.cc
ranger% aCC type.cpp queue.cpp eventList.cpp ola7.cc –o run7
ranger% run7
ranger% exit
```

Simulate()

```
Create an empty queue  bankQueue  to represent the bank line
Create an empty event list  eventList

Get the first arrival event from the input file
Place the arrival event in eventList

while (eventList is not empty)
{       newEvent = the first event in eventList

        if (newEvent is an arrival event)
                processArrival(newEvent, arrivalFile, eventList, bankQueue);
        else
                processDeparture(newEvent, eventList, bankQueue);
}
```

ProcessArrival(arrivalEvent, arrivalFile, anEventList, bankQueue)

```
atFront = bankQueue.isEmpty();

bankQueue.enqueue(arrivalEvent)
delete arrival event from anEventList

if (atFront)
{
        insert into anEventList a departure event that corresponds to the new
        customer and has currentTime = currentTime + transaction length
}

if (not at end of input file)
{
        get the next arrival event from arrivalFile
        add the event to anEventList
}
```

ProcessDeparture(departureEvent, anEventList, bankQueue)

```
bankQueue.dequeue();
Delete departureEvent from anEventList;

if(!bankQueue.isEmpty())
{       insert into anEventList a departure event that corresponds to the customer
        at the front of the line and has currentTime = currentTime + transaction length;
}
```