ECE 2574 ♦ Introduction to Data Structures and Algorithms
Homework 4: Event-driven Simulation

Overview: For this assignment, you must write a C++ program that simulates a line of customers at a bank. Your program must compute statistics to describe the length of time that customers are required to wait for service. This problem resembles many physical systems that make use of queues. Examples include communication networks, highway traffic, and assembly lines in factories.

Objectives: The main goals of this assignment are to gain experience in using queues and the Standard Template Library (STL), and in modeling real-world systems.

Program specifications: Write a C++ program that implements an event-driven simulation of a line of customers at a bank, similar to the description in Chapter 13 of the textbook. Each customer enters a queue, waits until reaching the front of the queue, performs a transaction with a teller (bank employee), and then leaves the bank. You may assume that the bank has only one queue and one teller.

A list of *arrival events* will be provided to your program from an input text file, similar to the following:

```
20  5
22  4
23  2
30  3
```

Each line of text represents a single bank customer, and contains a pair of integers. The first number is the *arrival time*, which is when that customer entered the bank queue; this is specified in minutes relative to the time when the bank opened for business at $t = 0$. The second number on each line is the *transaction time*, which is the amount of time (also in minutes) required for that customer's transaction. You may assume that the file's events are sorted in ascending order of arrival time.

The main goal of the program is to determine when *departure events* occur (i.e., when customers leave the bank), and compute statistics based on the length of time that customers spend in the bank. Your program needs to maintain a *queue of customers*, and an *event list*. As described in the textbook and lecture notes, the event list is an ADT that 1) keeps track of arrival and departure events that will occur but have not yet occurred, and 2) contains at most one arrival event and one departure event.

The output of the program should be sent to a text file. This output should provide a trace of events, along with statistics for the entire simulation run. For the input file shown above, the output should resemble the following:

```
t = 0:  Simulation begins
t = 20: Arrival event
t = 22: Arrival event
t = 23: Arrival event
t = 25: Departure event
t = 29: Departure event
t = 30: Arrival event
t = 31: Departure event
t = 34: Departure event

Total number of customers = 4
Average amount of time in the bank = 6.0 minutes
Maximum amount of time in the bank = 8.0 minutes
Average amount of time waiting = 2.5 minutes
Maximum amount of time waiting = 6.0 minutes
Maximum length of line = 3 customers
```

In the program's output, the "amount of time in the bank" should include a customer's transaction *and* waiting times. The "amount of time waiting" should *not* include a customer's transaction time. The "length of line" should include the customer who is engaged in a transaction.

You must use an STL `queue` to represent your line (i.e., queue) of customers. Use an ADT for your event list, either by selecting one from the STL (e.g., using the STL `list` container) or by implementing your own. The choice of ADT for the event list is up to you.

Your program should be invoked using the command line as follows:

```
simulate input_file output_file
```

Here, **`simulate`** is the name of your executable program, and `input_file` and `output_file` are the names of the input and output text files, respectively.

Exception handling: Your implementation should use `try`/`throw`/`catch` to handle the following exceptions:

1. If input/output files are not provided on the command line, or the input file does not exist.
2. If `new` returns `NULL` when allocating memory.
3. If the content of the input file is not in the expected format.

File naming conventions: Create a client source file named `bank.cpp` and place your event-driven simulation code in files named `simulate.h` and `simulate.cpp`.

What to hand in: Upload your source files (`bank.cpp`, `simulate.h`, and `simulate.cpp`) to Scholar. Be careful to use these file names, to test your files before uploading them, and to upload the correct files. The files that you upload to Scholar are the files that will be graded.

Partial credit: If you could not get your program to run, also turn in a written description (MS Word or PDF file) that describes 1) the problems you encountered, 2) what parts of your program work correctly, and 3) suggestions to the grader as to how your program may be tested for partial credit consideration. If your program runs while meeting all of the requirements described above, you do not need to turn in such a file.

Grading: The assignment will be graded according to the Grading Guidelines posted in Scholar.

Honor Code: You must work independently on this assignment. Please review the VT Honor Code statement in the syllabus.