Assignment 1—Due Monday, October 16

"The California DMV"

The good people of at the Department of Motor Vehicles (DMV) have a problem. They are concerned that people do not spend enough time waiting in lines to appreciate the privilege of owning an automobile. The current arrangement is as follows:

1. When people walk in the door, they must wait in line to **sign in**.
2. Once they have signed in, they are told either to stand in the line for **registration renewal**, or they are told to wait until they are called for **license renewal**.
3. Once they have completed their transaction, they must go and wait in line at the **cashier**.
4. When they finally get to the front of the cashier's line, if they expect to pay by check then they are told that all checks must be approved. To do this, they must go to the check-approver's table. After that they reenter the cashier's line at the very end.

Assume that the various transactions take the following amounts of time:

Sign in	20 seconds
Renew license	180 seconds
Renew registration	120 seconds
See cashier (pay by cash or approved check)	60 seconds
See cashier (present an unapproved check)	20 seconds

Assume that checks are approved instantly, and that people walk from one place to the next infinitely quickly. Therefore what happens upon arriving at the front of the cashier's line with an unapproved check is to wait for 20 seconds and then go the the rear of the cashier's line with a check that is now approved. As ridiculous as it may seem, the people waiting for license renewal are called in alphabetical order. However, no one is ever "pushed back" into line once his transaction has begun.

In this assignment you are to simulate the events which transpire at the DMV. Assume the following:

- Whether a customer comes in for license renewal or registration renewal is determined by flipping a fair coin $\{0, 1\}$.
- Whether a person pays by check or cash is determined by flipping a fair coin $\{0, 1\}$.
- All checks are approved.

- People arrive at random intervals. But on the average, there's a new person arriving every `mean_arrival_interval` seconds. Simulate the arrival time of the $(i+1)$-st person as follows: Start off by setting $s \leftarrow$ `mean_arrival_interval`. Now repeat the following until $b = 1$: Flip a fair coin $b \in \{0, 1\}$ and then, if $b = 0$, set $s' \leftarrow 2s'$. Now choose a random number in $\Delta \in [0, s')$ and assert that person $i+1$ arrives Δ seconds after person i arrived.

In this assignment you are to make a computer model to simulate the DMV office. Call your program `prog1`. Let `prog1` support command line arguments of `-mean_arrival_interval` and `−d`. The former has the obvious meaning, while the latter is used to display the queues each time that there is a change. Compute statistics which include the average amount of time that people's business takes.

Simulate one day (8 hours of the front door being open, and everyone eventually to be serviced) using different values of `mean_arrival_rate`. Approximate the *wait-10-capacity* of the DMV office, this quantity being defined as the maximal value of the `mean_arrival_rate` such that the average time for someone to complete their business is at most 10 minutes. Approximate the *wait-30-capacity* of the DMV office, defined analogously. Print out enough information to understand where people are getting "hung up." Explain your findings.

Turn in hardcopy of your source code; a `-d` execution (a short one, please!); an execution corresponding to the *wait-10-capacity*; and and a writeup. The writeup should be a 1–2 page high-level description of your solution (describing the ADTs used, their operations, their implementation, and why you made these choices). It should answer all of the questions asked above. Please highlight in your executions whatever is helpful to highlight.

In implementation, overload the "goes to" operator (`<<`) of the `ostream` class so that you can print queue information in a nice way. You should generally do this for all the classes which you might wish to output.

Sundry notes:

–To get a decent grade you must make appropriate use of C++ language features (especially classes).

–You can use `random` and `drand48` for generating your random numbers. A fragment of code to help you out will be posted to the newsgroup.

Hints: Start Early! Do toy problems in C++ first. Don't underestimate the amount of time to program reasonably in a new language, nor the complexity of this first program. Have fun!

—This assignment modified from *Data Abstraction and Problem Solving with C++*, Frank Carrano, Benjamin Cummings, 1995, pp. 350–51.