CX 4010 / CSE 6010 Assignment 2: Discrete Event Simulation

Due Dates:

- Due: 10:00 AM, Friday, October 3, 2014
- Revision (optional): 10:00 AM, Tuesday October 7, 2014
- No late submissions will be accepted

To complete this assignment, develop a discrete event simulation that models the operation of a "one-lane bridge." By this we mean there is a 2-lane road carrying traffic in two directions (say north and south) that crosses over a bridge. The bridge is only wide enough to carry traffic in one direction at a time, however. If the bridge is carrying traffic in one direction, say north, when a southbound vehicle arrives, that vehicle must wait until the direction of traffic crossing the bridge changes.

Assume it takes `C` seconds for a vehicle to cross the bridge. These `C` seconds includes `E` seconds (`E<C`) for the vehicle to "enter" the bridge, and `C-E` seconds to complete its journey to cross the bridge. For this assignment, assume `C` is 30 seconds, and `E` is 2 seconds, but your program should define `C` and `E` as constants that can be easily changed.

The rules for the direction of traffic crossing the bridge are as follows:

1. At any time, the bridge is in one of three states: `north`, `south`, or `empty`. If the state is `north` (`south`), there are one or more northbound (southbound) vehicles crossing the bridge. If the bridge is in the `empty` state there are no vehicles using the bridge.
2. If a vehicle traveling in either direction arrives at the bridge and the bridge is in the `empty` state, that vehicle may immediately enter and begin to cross the bridge, and the bridge state is set to the direction that vehicle is traveling.
3. If a northbound (southbound) vehicle arrives at the bridge and the bridge is in the `south` (`north`) state, the vehicle must stop and wait to use the bridge. If there are other vehicles already waiting to use the bridge, the vehicle must queue behind the other already waiting vehicles. When the bridge direction changes to the `north` (`south`) state, the vehicles will cross the bridge in the order in which they arrived. A collection of queued vehicles crossing the bridge one after the other in the same direction is referred to as a *group*.
4. When a group of vehicles begin to drive across the bridge, each vehicle in the group will enter the bridge, one after the other, `E` time units apart (i.e., if the bridge state changes at time t, the first vehicle will be on the bridge at time t+`E`, the second at t+2`E`, the third at t+3`E`, etc.) If a new vehicle arrives at the bridge traveling in the same direction as the group while at least one member of the group is still waiting to enter the bridge, that vehicle joins the group. If a vehicle arrives traveling in the same direction as a group crossing the bridge, but the last vehicle in the group has already entered the bridge, that vehicle must wait even though the state of the bridge is the same as the direction the vehicle is traveling.
5. When the last vehicle of a group completes its crossing of the bridge, then (1) if there are vehicles waiting to cross in the opposite direction, the state of the bridge changes to the opposite direction and the queued vehicles form a new group that begins crossing the bridge, or (2) if there are no vehicles waiting to cross in the opposite direction, but there are now vehicles waiting to cross in the same direction as the group that just left the

bridge, then the waiting group immediately begins crossing the bridge, or (3) if neither case (1) nor case (2) applies, the bridge enters the `empty` state.

Assume that the time between arrivals (the inter-arrival time) of northbound (southbound) vehicles at the bridge is drawn from an exponentially distributed random variable with mean `NB_InterArrivalTime` (`SB_InterArrivalTime`).

Your simulator should maintain statistics indicating the number vehicles crossing the bridge in each direction and the average waiting time encountered by northbound and the average waiting time for southbound vehicles. These statistics should be printed at the end of the run.

The simulation should begin at simulation time 0, and simulate the operation of the bridge for 4 hours.

You must follow the following rules concerning the implementation:

1. Use a double precision floating point number to represent simulation time.
2. Storage for events must be allocated dynamically by calling `malloc(),` and the storage released when you are done using the memory by calling `free()`. Programs that have memory leaks or dangling pointers will be considered erroneous!
3. Your simulator must implement the future event list as a priority queue constructed as a linear linked list sorted according to simulation time.
4. Each event must be implemented as a C `struct`. It must include a timestamp value, any parameters you deem necessary to characterize the event, and a pointer to the function that is called to process the event.
5. To generate random numbers from an exponential distribution with mean `U`, create a function `double urand(void)` that returns a random number uniformly distributed over the interval [0,1) (note it cannot return the value 1.0), then define `double randexp()` that returns `-U*(log(1.0—urand()))` where `log()` is the C function defined in `<math.h>` to compute a natural logarithm.

This assignment consists of three parts. The first part involves developing a priority queue and the main loop of the simulation for processing events, and generating (and processing) a stream of arrival events (but not actually simulating the bridge) to test the program. You must design the priority queue as a distinct "module," with a well-defined interface that should include functions to (1) create a new (empty) priority queue, (2) insert a timestamped event, and (3) remove the smallest timestamped event from the queue. You need not turn in the results from part (1) but it is strongly recommend you develop this first to help you develop the software in an incremental fashion.

The second part involves developing the simulation for the one-lane bridge described above. The third part involves running the simulation with different parameters to collect statistics and writing up your results.

1 Student Taking CX 4010

If you are a CX 4010 student, you must:

1. Implement the simulation and demonstrate to the TA that your program is functioning correctly. To accomplish this, generate a short simulation and print out a timestamp-ordered series indicating when each vehicle arrives at the bridge (assign each vehicle an

integer ID to distinguish one vehicle from another), when it enters the bridge to begin to cross, and when it leaves the bridge.

2. Complete a series of runs increasing the arrival rate (1 / interarrival time) assuming the arrival rates of vehicles in each direction is the same. Show a graph with two lines indicating the average waiting time experienced by vehicles traveling in each direction as a function of the arrival rate. Vary the arrival rate to fully explain the behavior of this system for different traffic loads. In your report explain any anomalies or unexpected results in your plot. Pay special attention to your results when the arrival rate becomes large.

2 Students Taking CSE 6010

If you are a CSE 6010 student you must:

1. Complete the steps described above for students taking CX 4010.
2. Perform a series of experiments evaluating the performance of your priority queue as a function of the number of elements in the queue. Write a test program when the queue initially contains N values, then repeatedly (1) remove the smallest timestamped event, and (2) generate and insert a new event with timestamp equal to that of the event just removed from the event list plus an exponentially distributed random value with mean of 1.0. Each cycle through your loop including a remove and insert operation is called a "hold" operation. Perform a large number of hold operations, and compute the average time to perform one hold operation. Plot the time to perform a hold operation as a function of the number of events in the event list, and explain your results. Are they consistent with what you would expect from a Big-O analysis of the program?
3. Modify your simulator to include the following rule: the maximum number of vehicles in a group that are allowed to cross the bridge at one time is 10 vehicles. If a queue contains, say 15, vehicles, when the bridge becomes available for traffic in this direction, only 10 vehicles are allowed to cross the bridge, and the remaining 5 vehicles must wait until the next time vehicles in that direction are allowed to cross.
4. Repeat the set of experiments varying the arrival rate, and again plot your results. Explain your results, and compare them to what you obtained without the addition of the above rule.

3 All Students

Turn in your report including your results, and all software in a single zip file. Your software must be well documented and include comments so the code is easy to understand. You should include a README file with instructions on how to compile and run your program on the jinx cluster. Finally, please note that all code you turn in must be completely developed by yourself, although we encourage you to discuss the problem and issues you are facing with other students (as well as the TA/instructor).