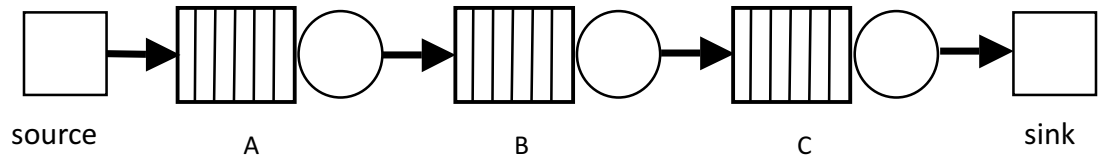
CX 4010 / CSE 6010 Assignment 2: Queueing Network Simulation

Due Dates:

- Due: 10:00 AM, Friday, September 30, 2016
- Revision (optional): 11:55 PM, Monday October 3, 2016
- No late submissions will be accepted

1 Discrete Event Simulation Application

To complete this assignment, develop a discrete event (as opposed to a time-stepped) simulation that models a simple manufacturing system. This simulation is known as a queueing model in the literature. Queueing models are widely studied and used to model systems such as customers using a facility (e.g., a department store or restaurant), air and road transportation systems, computer networks and computer systems, and manufacturing systems, to mention a few. All of these systems involve *customers* (e.g., people, aircraft, vehicles, data packets, computer jobs, parts) that must travel from one *station* to another and receive "service" or "processing" at each station. Each station can only process one of these entities (e.g., a customer) at a time. Customers wanting to use a facility that is busy handling another customer must wait in a queue until the station is ready to process the next customer.



Here, you will simulate the operation of a simple manufacturing system represented as a "tandem queue" as shown above. Specifically, the system of interest is an assembly line where parts are generated at a source, and move through three stations, A, B, and C in succession. Parts leaving station C move to a "sink" where they exit the system. At any instant, a station is either busy processing a part or is idle. If the station is idle when the part arrives, it begins processing the part. If the station is busy, the part is placed in a queue where it will wait until the station can process it. Parts are removed from the queue in first-in-first-out (FIFO) order.

Each part maintains certain attributes as it travels through the system. Specifically, when created each part is assigned an attribute indicating the amount of time required to service the part at each station (this service time is the same for all three stations). The service time should be selected from an exponential distribution with mean value of S. Assume that the time between the generation of successive parts at the source, i.e., the interarrival time, is also exponentially distributed with mean A.

Your simulation should compute the average amount of time a part remains in the system as well as the average of the total waiting time experienced by each part in traveling through the three stations. These values should be printed at the end of the simulation run. In your experiments, assume the average service is 10.0 units of time, and run your simulation for 10,000 units of simulation time.

Your software must adhere to the following rules concerning the implementation:

1. You must implement an event-driven, discrete event simulation, *not* a time-stepped simulation.
2. Use a double precision floating point number to represent simulation time.

3. 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!
4. Similarly, `malloc()` must be used for each part to create storage that holds information concerning that part. This storage should be allocated when the part is created at the source and released when the part reaches the sink.
5. Your simulator must implement the future event list as a priority queue constructed as a linear linked list sorted according to simulation time.
6. Each event and part must be implemented using a C `struct`. Each event must include a timestamp value and any parameters you deem necessary to characterize the event. Each part will have attributes and statistics associated with the part stored in the structure.
7. 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.

The software is composed of two parts. You will work in teams of two students each to complete this assignment, with one student responsible for implementing each part. The first part involves developing a software library that will used to create discrete event simulations in general, independent of the particular application. It should include a priority queue, a library module implementing first-come-first-serve (FIFO) queues, and a random number generation function.

The second part includes the main processing loop for the discrete event simulation that repeatedly removes the smallest time-stamped event and calls an event handler function to process that event. This code also should also print the results of the simulation once the execution has been completed.

The interface between the two parts of the simulation must be clearly documented in a `.h` file. This file should include only that information necessary to use the software module, and provide no information concerning details of the internal implementation of the module.

Your software must be well modularized with a clear application program interface (API) defined and documented in your code. The programmer using the software modules you create should not need to know anything about how you implemented these functions, only the interface for using them. Specifically, 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. Similarly, the FIFO queue should have functions to (1) create a new FIFO queue, (2) add an item to the queue, and (3) remove the next item from the queue.

2 Student Taking CX 4010

If you are a CX 4010 student, you must:

1. Implement the simulation and generate a print out to demonstrate to the TA that your program is functioning correctly. To accomplish this, generate a short simulation and print out a series indicating what events are processed in the simulation, and argue why it is correct.

2. Complete a series of runs increasing the arrival rate (1 / A). Vary the arrival rate to fully explain the behavior of this system for different workloads. Generate a graph showing the average time in the system and queueing delay of parts as a function of the arrival rate. In your report explain any anomalies or unexpected results in your plot. Pay special attention to your results when the arrival rate becomes large.
3. 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.

3 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. Examine the literature on queueing networks to determine what waiting time should be obtained for the case of a *single* queue/server. Create a simulation for a single queue, and compare your results with the theoretical results that should be obtained by plotting both on a single graph. Write up your results, including references to the relevant literature.
3. Write a program to measure the average time to perform a single insert followed by a single remove operation in the priority queue containing N events where each event has a timestamp uniformly distributed between 0.0 and 1.0. The new event should have a timestamp also uniformly distributed over the same interval. Show a graph indicating the execution time as N increases. This program should follow the following steps:

```
Initialize the priority queue to contain N events
start timer
loop a large number of times
        insert into priority queue
        delete from priority queue
endloop
stop timer
print average time for one iteration of the loop
```

4 CX 4010 Students (Extra Credit)

Complete the assignment required for CSE 6010 students described above.

5 Reminder: Collaboration Policy

A reminder you must adhere to the Georgia Tech honor code, and the collaboration policy stated in the course syllabus. Specifically, you are encouraged to discuss the problem and possible solutions with other students (as well as the TA/instructor), however, all code that you turn in must be completely your own work. Disseminating your code to other students (not in your team) is strictly prohibited. Further, downloading code from the web or other sources other than examples provided in class for use in this assignment is also prohibited.