CSE 605 Performance Evaluation
Midterm Exam
Fall 2003

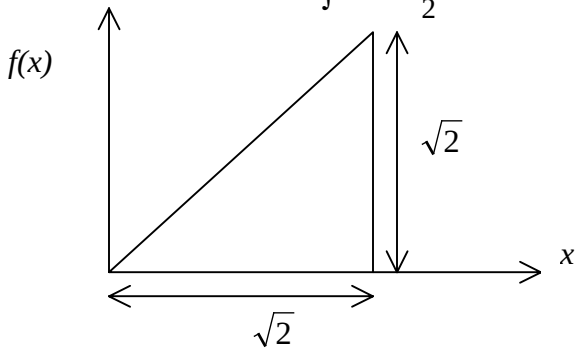
Name:

1. A database system consists of one CPU and five disks. The system is measured for 100 seconds. The measurements show a total of 1000 transactions completed, a CPU utilization of 50%, and utilizations of 40%, 25%, 20%, 10%, and 5% for each of the five disks. It is also measured that on average there are 3 transactions being processed inside the system at one time (this means that there are an average of 3 transactions total in the CPU and in the disks – either in service or waiting in a queue). Determine what fraction of an average transaction's overall response time is spent waiting for service anywhere inside the system?

Total utilization of all devices = 0.5+ 0.4+0.25+0.2+0.1+0,05 = 1.5.
This means that there on average 1.5 transactions at the servers.
But we know that there on average 3 transactions in the system.
So, there must be 3-1.5 = 1.5 transactions on average waiting in one or more queues.
Thus, Q/L = 0.5. Apply Little's law to the whole system. Q = W*λ and L = R* λ. Thus, W/R = Q/L = 0.5.

The information about throughput is superfluous.

2. Use the inverse transform technique to show how you will generate random number samples from a distribution as described by the following pdf *f(x)*, given that you have access to random numbers *U* from a uniform distribution between 0.0 and 1.0. [Helpful formula: $\int x dx = \frac{x^2}{2}$]



Note that $f(x) = x$. Thus, $F(x) = x^2/2$. Set $F(x) = u$. Solve for u to get $x = \sqrt{2u}$, where u is a random number from uniform distribution between 0.0 and 1.0.

3. Suppose, our department is thinking of upgrading the departmental servers and computer network, and you are playing the role of the system architect. You have made a queuing network diagram of the various components of the system; evaluated the average user behavior (i.e., average number of requests per unit time that a user generates) and also evaluated the visit ratios of different components from extensive instrumentations. You also know the service times of different components (both current and upgraded versions) from the manufacturer data sheet or your own instrumentation.

   Is this information sufficient for you to carry out a bottleneck analysis and determine which component(s) require upgrading for the number of users to be supported, or what is the critical number of users that can be reasonably supported with the current system?

   Is this information sufficient for you to determine the average response time for a request for the current or upgraded network? If yes, explain how you can determine the response time. If not, what other information you will need.

4. Consider the single queue simulation discussed in the class (code reproduced below). Assume that the server now takes a break as soon as the queue becomes empty and does not come back until there are *k* customers waiting.

   How will you incorporate this break in the simulation program? [Present the basic idea in a few sentences and then modify the pseudo-code provided. Simply supply the changes. No need to re-write the entire code.]

No change needed for departure. For arrival, only change needed should reflect that server does not become busy (from idle) until k customers accumulate in the queue. The change in the code is in bold.

```
We have used two type of events : Arrival(), noted as A, and
Departure(), noted as D.

We have used two state variables: queue_length (no. of customers
in queue, not including the one currently in service, if any,
initialized to 0) and server (idle or busy, initialized to idle).

Event handler routines follow:

Arrival() {
  new_ev.type = A;
  new_ev.ts = Clock + random1(); /* random1() is for inter-arrival time
*/
  Schedule(new_ev);
  if ((server == busy) || (queue_length < k)){ /* wait */
     queue_length++;
```

```
  } else { /* begin service */
     server = busy;
     new_ev.type = D;
     new_ev.ts = Clock + random2(); /* random2() is for service time */
     Schedule(new_ev); /* schedule own departure */
  }
}

Departure() {
  if (queue_length == 0) /* this is the last customer */
     server = idle;
  else {
     queue_length--;
     new_ev.type = D;
     new_ev.ts = Clock + random2(); /* random2() is for service time */
     Schedule(new_ev); /* schedule next departure */
  }
}
```