Mid-Term Exam for *Simulation* (CIS 4930) Summer 2009

>>> SOLUTIONS <<<

Welcome to the Mid-Term Exam for *Simulation* (CIS 4930). Read each problem carefully. There are ten required problems (each worth 10 points). There is also an additional extra credit question worth 10 points. You may have with you a calculator, pencils and/or pens, erasers, blank paper, and one 8.5 x 11 inch "formula sheet". On this formula sheet you may have anything you want (definitions, formulas, homework answers, old exam answers, etc.) as **handwritten by you in pencil or ink** on both sides of the sheet. Photocopies, scans, or computer generated and/or printed text are not allowed on this sheet. Note to tablet PC users – you may **not** print-out your handwritten text for the formula sheet. You have 120 minutes for the exam. **Please use a separate sheet of paper for each question**. Good luck and be sure to show your work!

Problem #1

2 pts for each sub-problem.

Answer the following questions regarding the basics systems and modeling:

a) What is a system? Give a short formal definition. (Hint: "what you draw a box around" is not a formal definition)

A system is a set of interacting or interdependent entities forming an integrated whole.

b) What are the four possible ways to study a system?

We can 1) do a mathematical analysis, 2) build a simulation model and experiment on it, 3) build a prototype system and experiment on it, and 4) build the actual system and experiment on it.

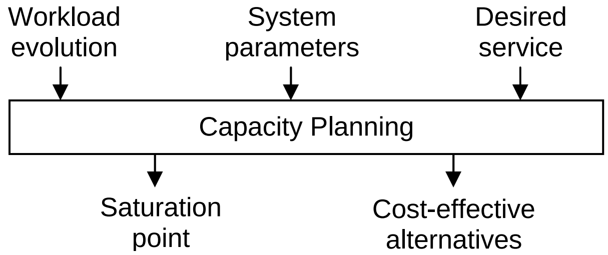
c) What is a model? Give a short formal definition.

"A model is a representation (physical, logical, or functional) that mimics another object under study." (Molloy 1989).

d) What is computer simulation? Give a short formal definition.

"Computer simulation is the discipline of designing a model of an actual or theoretical physical system, executing the model of a computer, and analyzing the execution output." (Fishwick, 1995)

e) What are the inputs to the capacity planning process? What are the outputs? (Hint: A diagram might be helpful)



Problem #2

2 pts for each sub-problem.

Answer the following questions regarding performance:

a) What is performance? Give a short formal definition.

"Performance is the quantitative measure of a system."

b) What are the two most common performance measures for an ICT system?

Throughput and delay

c) Give the formulas for speed-up and relative change for two systems.

If we have rate R1 for system #1 and R2 for system #2 and if completion time T1 for system #1 and T2 for system #2, then speed-up of system #2 with respect to system #1 = R2 / R1 = T1 / T2. Relative change of system #2 with respect to system #1 = (R2 – R1) / R1 = (T1 – T2) / T2.

d) What is a factor (in the context of experiments)? Give an example of a factor with multiple levels (and identify what some of the levels might be).

A factor is a variable that affects the response of a system. Another name for factor is control variable. The amount of RAM in a PC is a factor where the levels could be 1 GB, 2 GB, 4 GB, and so on.

e) What is the goal in the design of experiments?

To determine the maximum amount of information (about a system) with the least amount of effort.

Problem #3

3 pts for (a), 3 pts for (b), and 4 pts for (c) (partial credit if at least get p^3 right).

Answer the following questions regarding probability theory:

a) What is the experimental definition of probability?

$$\Pr[\text{outcome}] = \lim_{n \to \infty} \frac{\text{Number of observed outcomes}}{n \text{ repetitions of experiment}}$$

b) What does it mean for events to be independent?

For independent events, the occurrence of one event does not make it more or less probable that some other event occurs.

c) Assume that you have three computers and two sensors in a subway train. If all three computers and one or both sensors fail, then the subway train is uncontrolled. If the probability of a computer failing is p and the probability of a sensor failing is q at any given time interval, what is the probability of the subway train becoming uncontrolled in any given time interval? Assume that failures are independent.

All failure events are independent, so Pr[train uncontrolled in an interval] = $p^3(1 - (1 - q)^2)$. The first term is the probability of all three computers failing. The second term is the probability of 1 or more sensors failing. The second term is derived as follows $(1 - q)$ is the probability of a sensor being up, so $(1 - q)^2$ is the probability of both sensors being up. Then $(1 - (1 - q)^2)$ is the probability of both sensors being not up (i.e., one or two have failed).

Problem #4

2 pts for (a), 2 pts for (b), 3 pts for (c), and 3 pts for (d)

Answer the following questions regarding probability theory (specific to random variables and distributions):

a) What is a random variable?

A random variable is a function that maps a real number to every possible outcome in the sample space.

b) What is a heavy tailed distribution (i.e., what are its key properties or characteristics)? Give an example of a real-life event (related to ICT systems) that may be heavy tailed.

A heavy-tailed distribution has most of its mass in its tail. Often, a heavy tailed distribution will have infinite mean and/or variance. Download sizes may be heavy-tailed. Most downloads are small, but a very few are extremely large (say, network back-ups of an entire hard drive).

c) Define *X* to be a random variable that takes on the value of the number of failed servers for a given period of observation. Over many observation periods you have found the probability that 0 servers fail in a given period of observation is 0.5, 1 server fails is 0.2, and 2 servers fail is 0.3. What is the mean value of *X*? What is the standard deviation of *X*? Plot the pmf and CDF of *X*.

Pr[X = 0] = 0.5, Pr[X = 1] = 0.2, and Pr[X = 2] = 0.30.

E[X] = (0.5)(0) + (0.2)(1) + (0.3)(2) = 0.80

Variance = $E[X^2] - E[X]^2 = ((0.5)(0)^2 + (0.2)(1)^2 + (0.3)(2)^2) - (0.8)^2 = 0.76$

```
f(x) |
 1.0 +
     |
 0.8 |
     |
 0.6 +
     o
 0.4 +
     |       o
 0.2 +   o   |
     |   |   |
     +---+---+-----
     0   1   2   x
```

```
F(x) |
 1.0 +       +-----
     |       |
 0.8 |       |
     |   +---+
 0.6 +   |
     +---+
 0.4 +
     |
 0.2 +
     |
     +---+---+-----
     0   1   2   x
```

d) Assume that you have an old “buggy” Pentium processor. The probability, *p*, of a given calculation failing is 1 in 100,000. Failed calculations are independent. Given 100,000 calculations, what is the probability that 1 calculation will have failed? Show your work.

This is a binomial distribution. So, we solve for (where p = 1/100000 = 10^{-5})

$$\mathrm{combin}(100000, 1) \cdot p^1 \cdot (1-p)^{100000-1} = 0.367881$$

We note that N choose 1 is always N. We do not need to solve for any really large factorials. We could also note that p is small and N is large so we can approximate this as a Poisson distribution with $\lambda = N*p = 1$. Then:

$$\frac{\lambda^k}{k!} \cdot e^{-\lambda} = 0.367879$$

Problem #5

5 pts for each sub-problem. For (a) 3 pts for formula, 2 pts for code around it. For (b) 3 pts for concept, 2 pts for code around it.

Answer the following questions regarding workload. For both questions you may assume that you have a function `randUnif()` that returns a uniformly distributed random value between 0 and 1.

a) Write a C function that returns exponentially distributed random variables with mean value *x*.

```
double exponential(double x)
{
  double z;          // Uniform random number (0 < z < 1)
  double exp_value;  // Computed exponential value to be returned

  // Pull a uniform random number (0 < z <= 1)
do
{
  z = randUnif();
} while (z == 0.0);

  // Compute exponential random variable using inversion method
  exp_value = -x * log(z);

  return(exp_value);
}
```

b) Assume you have made measurements on the size of file downloads. You have observed 1 million file downloads and found that 900,000 downloads were exactly 10 KBytes, 199,999 downloads were exactly 100 Kbytes, and 1 download was exactly 1 GByte. Download sizes are independent of each other. Write a C function that returns a download size based on an empirical distribution from the measurements made.

Oops. The sum of measurements is 1.1 million and not 1 million. This error was announced during the exam. Two possible solutions are to assume 1.1 million as the total measurements or use 99,999 as the number of 100 Kbyte files measured. The solution here assumes the latter.

```
int downloadSize()
{
  double z;

  // Pull a uniform random number (0 <= z <= 1)
  z = randUnif();

  // Return a download size based on an empirical distribution
  if (z < ((double) 900000 / 1000000)) return(10*1024);
  if (z < ((double) 999999 / 1000000)) return(100*1024);
  return(1024*1024*1024);
}
```

Problem #6

3 pts for (a) (1 pt for each method), 3 pts for (b) (1 pt for each method), and 4 pts for (c) (2 pts for histogram, 1 pt for autoc, and 1 pt for 2D graph).

Answer the following questions regards random number generation.

a) What are the desired properties of an RNG?

The four desired properties are 1) return uniformly distributed and independent values, 2) be fast and not require much storage, 3) stream of values should be reproducible, and 4) provision for multiple independent streams.

b) What methods are there for generating random numbers?

Three methods are: 1) Use a physical system known to produce random values (e.g., radioactive decay), 2) Use a table of known good random values (e.g., generated from a physical system), and 3) use an algorithm.

c) Briefly, how can an RNG be tested for "goodness"?

An RNG should be tested for a 1) uniform distribution by plotting a histogram of values – the histogram should show a uniform distribution, and 2) independence of values. Independence can be partially tested by looking for non-zero autocorrelation (autocorrelation tests for linear dependence). A better test is a 2D plot of value pairs and an "eyeball" look for patterns.

Problem #7

2 pts for each sub-problem.

Answer the following questions about queueing.

a) What are the key characteristics of queues that we are interested in?

Key characteristics are arrival process, service time distribution, number of servers, system capacity, population size, and service discipline.

b) What is Kendall notation. Describe it.

Kendall notation describes a queue as A/S/c/k/m where A is the arrival process, S is the service time distribution, c is the number of servers, k is the system capacity, and m is the customer population. A and S can be M for Markov (exponential), D for deterministic, and G for general. A category with a numerical quantity that is infinity is omitted.

c) State Little's Law.

L = λW where L = number of customers in the system, λ = arrival rate of customers, and W = wait time (this includes service time) of a customer in the system.

d) Given an M/M/1 queue with utilization of 90%, what is the mean number of customers in the system? What is the mean number of customers in the queueing area?

$$L = \frac{\rho}{1-\rho} = \frac{0.9}{1-0.9} = 9 \text{ customers}$$

Lq = *L* – ρ, so Lq = 8.1 customers.

e) Repeat (d) for an M/D/1 queue.

We have to use the P-K formula and know that $C_s^2 = 0$ for deterministic service time. The P-K formula is:

$$L = \rho + \frac{\rho^2\left(1 + C_s^2\right)}{2(1-\rho)}$$

So,

$$0.9 + \frac{0.9^2(1+0)}{2(1-0.9)} = 4.95 \text{ customers}$$

Problem #8

5 pts for scheduling a vacation, 5 pts for modified scheduling of service due to a current vacation.

Appendix A contains the source code for a discrete-event simulation model of an M/M/1 queue (the code for the functions `rand_val()` and `exponential()` is not shown). Change the source code to model an M/M/1 queue that takes a "vacation" whenever the system empties out (i.e., has no customers in it). A vacation is of length `T_VAC` and no customer service (should a customer arrive during the vacation period) will occur during a vacation. That is, if a customer arrives during a vacation its time in the queueing system will be the remaining vacation time plus its normal service time.

The solution is in Appendix A. New and changed code is highlighted in yellow.

Problem #9

4 pts for figure, 1 pt for each formula and calculation. No double jeopardy if figure is wrong.

Consider the following single-server queueing system from time = 0 to time = 20 sec. Arrivals and service times are:

- Customer #1 arrives at t = 1 second and requires 5 seconds of service time
- Customer #2 arrives at t = 1 second and requires 2 seconds of service time
- Customer #3 arrives at t = 2 seconds and requires 3 seconds of service time
- Customer #4 arrives at t = 12 seconds and requires 6 seconds of service time

Solve for system throughput (X), total busy time (B), mean service time (Ts), utilization (U), mean system time (delay in system) (W), and mean number in the system (L). Show your work to receive full credit.

```
Num in sys |
           |
         3 +     +-----------+
           |     |           |
         2 +  +--+           +-----+
           |  |                    |
         1 +  |                    +--------+  +-----------------+
           |  |                             |  |                 |
           +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+----
              1  2  3  4  5  6  7  8  9 10    12    14    16    18    20  time
```

X = C/T = 4/ 20 = 0.2 cust/sec
B = 16 seconds (by inspection)
Ts = B/C = 16 / 4 = 4.0 sec / cust
U = B/T = 16/20 = 80%
W = Sum of w / C = 27 / 4 = 6.75 sec
L = Sum of w / T = 27 / 20 = 1.35 cust

The previously posted exam had B = 17, the correct answer is B = 16 (by inspection). This affects the calculation of Ts and U.

Problem #10

2 pts for main loop, 2 pts for determine NextEvent and time assignment, 2 pts for event routines with switch, 2 pts for done test, and 2 pts for report.

Sketch the flowchart (or give pseudocode) of the key steps in a Discrete Event Simulation. Explain what is an event list, what it contains, and where it comes into play in the flowchart. Also explain what an event routine does.

```
1.  initialize
2.  do
3.     determine NextEvent
4.     time = nextEvent.time
5.     switch (nextEvent.eventNumber)
6.     case 1:
7.         Event routine for event 1
8.         Generate next event(s)
9.     case 2:
10.        Etc. (for all possible eventNumber)
11.    Update statistics
12. until done condition is met
13. output report
```

The event list is a linked-list in time order of events. An event is a structure with at least two members – time of event and event number. The event list comes into play in line 3 (dequeEvent() and line 8 (insertEvent()). Event routines change system state (that is change the values of the state variables).

Extra Credit Problem

4 pts for key variables, 4 pts for main loop, 2 pts for final calculation of area from hitCount.

Write a Monte Carlo simulation to integrate sin(x) from 0 to π (that is, find the area under sin(x) from 0 to π). To five digits, π = 3.14159. Note that C does include a `sin()` function in the math library. Focus on the main program. You may assume that you have a function named `randUnif()`that returns a uniformly distributed random value between 0 and 1.

```
#include <stdio.h>
#include <math.h>

#define NUM_ITER  1000000        // Number of iterations to run for

double rand_val(int seed);       // RNG for uniform(0.0, 1.0) from Jain

int main()
{
  double   x, y;                 // X and Y values
  double   x_max;                // Integration bound for x for f(x)
  double   y_max;                // Integration bound for y for f(x)
  double   hitCount;             // Count of hits within the quarter circle
  double   area_est;             // Estimated area
  int      i;                    // Loop counter

  // Initialize x_max and y_max
  x_max = 3.14159;
  y_max = 1;

  // Seed the RNG and initialize hitCount to zero
  rand_val(1);
  hitCount = 0.0;

  // Do for NUM_ITER iterations
  for (i=0; i<NUM_ITER; i++)
  {
    // Throw the dart
    x = rand_val(0) * x_max;
    y = rand_val(0) * y_max;

    // Determine if the dart is under the sin(x) curve
    if (y < sin(x)) hitCount++;
  }

  // Estimate the area
  area_est = (hitCount / 1000000) * (x_max * y_max);

  // Output results
  printf("Estimated area (from 0 to %f) = %f \n", x_max, area_est);

  return(0);
}
```

Appendix A – Source code for a discrete event simulation model of an M/M/1 queue

```
//==================================================== file = mm1_simple.c =====
//=  A simple "straight C" M/M/1 queue simulation                              =
//=   - No statistics gathering or reporting                                   =
//==============================================================================
<SNIP SNIP>

//----- Include files ----------------------------------------------------------
#include <stdio.h>              // Needed for printf()
#include <stdlib.h>             // Needed for exit() and rand()
#include <math.h>               // Needed for log()

//----- Constants --------------------------------------------------------------
#define SIM_TIME   1.0e6        // Simulation time
#define ARR_TIME   1.25         // Mean time between arrivals
#define SERV_TIME  1.00         // Mean service time
#define T_VAC      2.00         // Define vacation time

//----- Function prototypes ----------------------------------------------------
double rand_val(int seed);      // RNG for unif(0,1)
double exponential(double x);   // Generate exponential RV with mean x

//===== Main program ===========================================================
int main(void)
{
  double   end_time = SIM_TIME; // Total time to simulate
  double   Ta = ARR_TIME;       // Mean time between arrivals
  double   Ts = SERV_TIME;      // Mean service time
  double   time = 0.0;          // Simulation time
  double   t1 = 0.0;            // Time for next event #1 (arrival)
  double   t2 = SIM_TIME;       // Time for next event #2 (departure)
  double   t3 = 0.0;            // Time for current vacation to end
  unsigned int n = 0;           // Number of customers in the system

  // Seed the RNG
  rand_val(1);

  // Main simulation loop
  while (time < end_time)
  {
    if (t1 < t2)                     //** Event #1 (arrival)
    {
      time = t1;                     // Set time to that of current event
      n++;                           // Increment number of customers in system
      t1 = time + exponential(Ta);   // Assign time for the next arrival event
      if (n == 1)                    // If first customer in system
      {
        if (time < t3)
          t2 = t3 + exponential(Ts);
        else
          t2 = time + exponential(Ts);
      }
    }
    else                              // *** Event #2 (departure)
    {
      time = t2;                      // Set time to that of current event
      n--;                            // Decrement number of customers in system
      if (n > 0)                      // If customers in system then
        t2 = time + exponential(Ts);  //   assign next departure time
      else                            // If no customers in system then
      {
        t2 = end_time;                //   assign next departure to "infinity"
        t3 = time + T_VAC;            //   set vacation time to end at t3
      }
    }
  }
  return(0);
}
```