

>>> SOLUTION <<<

Welcome to exam #1 in *Computer Simulation*. Read each problem carefully. There are eight required problems (each worth 12.5 points) and one extra credit problem worth 10 points. Please write your answers in the space provided. You may have a calculator and one 8.5 x 11 inch sheet of paper with you. On this sheet you may have anything you want (definitions, formulas, flow charts, etc.) in your handwriting on both sides of the page. You may not have photocopies or printed text on your formula sheet. Please answer each problem on a separate sheet of paper. **Good luck!!!**

Problem #1

a) Precisely define “model” and give three reasons why we build and experiment with models.

A model is a representation (physical, logical, or functional) that mimics another object under study. The goal in building a model is to be able to predict the behavior of the original object. Building and studying models is usually cheaper and safer than studying the real object. Sometimes it is not possible to study the actual object (when it does not exist) and a model is thus required.

b) Describe capacity planning. What are the inputs to the process and what are the outputs?

Capacity planning is a process where the inputs are workload evolution, system parameters, and desired service and the outputs are saturation point and cost-effective alternatives.

c) Define computer simulation.

Computer simulation is the discipline of designing a model of an actual or theoretical physical system, executing the model on a computer, and analyzing the execution results.

d) What are some of the key performance measures of interest in IT systems?

Response time or delay is the key measure of interest for most IT systems. Throughput, loss, utilization, reliability, and availability are other measures of interest.

Problem #2

You are responsible for the operation of three server computers. Every morning you check each server if it is “up” or “down”. The probability of any given server being “down” is $p = 0.10$ (and this is independent of the state of the other servers). Solve for $\Pr[i$ servers are down] for $i = 0, 1, 2,$ and 3 . What is the mean number of servers that are down in the morning? What is the variance? What is the standard deviation?

This is a binomial distribution. We solve for

$$\Pr[i = 0] = \binom{3}{0} (0.10)^0 (0.90)^{3-0} = 0.729 \quad \text{mean} = n \cdot p = 3 \cdot 0.10 = 0.30$$

$$\Pr[i = 1] = \binom{3}{1} (0.10)^1 (0.90)^{3-1} = 0.243 \quad \text{variance} = n \cdot p \cdot q = 3 \cdot 0.10 \cdot 0.90 = 0.27$$

$$\Pr[i = 2] = \binom{3}{2} (0.10)^2 (0.90)^{3-2} = 0.027 \quad \text{standard deviation} = \sqrt{n \cdot p \cdot q} = 0.5196$$

$$\Pr[i = 3] = \binom{3}{3} (0.10)^3 (0.90)^{3-3} = 0.001$$

Problem #3

You are given the following transaction trace data (e.g., from a database server) where 11:59:58 is time zero.

Time	Transaction size
11:59:58	100 bytes
12:00:00	200
12:00:01	100
12:00:04	200
12:00:10	300
12:00:13	300

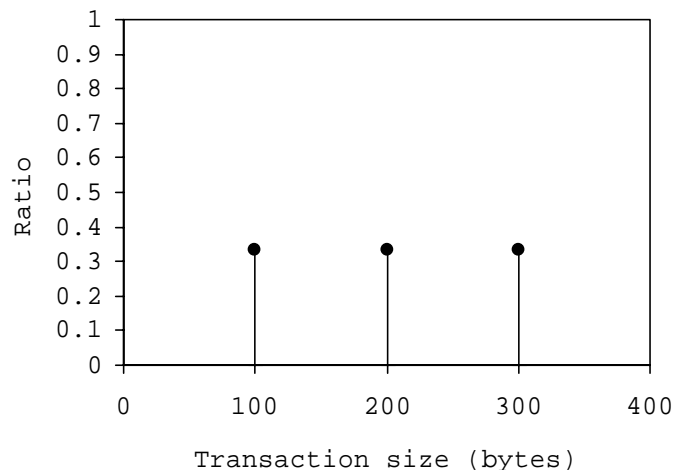
a) What is the mean interarrival time? What is the mean transaction size?

The interarrival times are 2, 1, 3, 6, and 2 seconds. The mean is thus $\frac{1}{5}(2+1+3+6+3)=3$ seconds. The transactions sizes are 100, 200, 100, 200, 300, and 300 bytes. The mean is thus $\frac{1}{6}(100+200+100+200+300+300)=200$ bytes.

b) What is the standard deviation of transaction size?

$$Standard_deviation = \sqrt{\frac{1}{6}((100-200)^2 + (200-200)^2 + (100-200)^2 + (200-200)^2 + (300-200)^2 + (300-200)^2)} = 81.65$$

c) Plot a histogram of the transaction size.



d) Write a C function that returns a value that is empirically distributed based on the trace data for transaction size. You may assume that you can call a function `rand_val()` that will return a uniformly distributed random variable between 0 and 1.

The function is:

```
double emp_transaction_size(void)
{
    double z;

    z = rand_val();
    if (z < (1.0 / 3.0)) return(100.0);
    if (z < (2.0 / 3.0)) return (200.0);
    return (300.0);
}
```

Problem #4

Write a C function that will return a Pareto distributed random variable with parameters a and b . The CDF of a Pareto distribution is $F(x) = 1 - \left(\frac{b}{x}\right)^a$. You may assume that you can call a function `rand_val()` that will return a uniformly distributed random variable between 0 and 1.

First we need to invert the function to solve for x as follows:

$$z = 1 - \left(\frac{b}{x}\right)^a$$

$$\left(\frac{b}{x}\right)^a = 1 - z$$

$$x = \frac{b}{\sqrt[a]{1-z}} = \frac{b}{\sqrt[a]{z}}$$

Then we can write the C function as:

```
double pareto(double a, double b)
{
    double z;

    do
    {
        z = rand_val();
    } while (z == 0.0);

    return(b / pow(z, (1.0 / a)));
}
```

Problem #5

Answer the following questions about queueing theory:

a) Describe Kendall notation

A queue is described as A/S/c/k/m where A is the arrival distribution, S is the service distribution, c is the number of servers, k is the capacity of the system in number of customers, and m is the number of customers in the universe. A and S can be "M" for Markov, "D" for deterministic, "G" for general, and others.

b) State Little's Law

$L = \lambda \cdot W$ for L is the mean number in the system, λ is the arrival rate, and W is the mean wait in the system

c) Given λ (arrival rate), μ (service rate), L (mean number in the system), and W (mean wait in the system) show how to solve for Lq (mean number in the queue), and Wq (mean wait in the queue) for a single server queue.

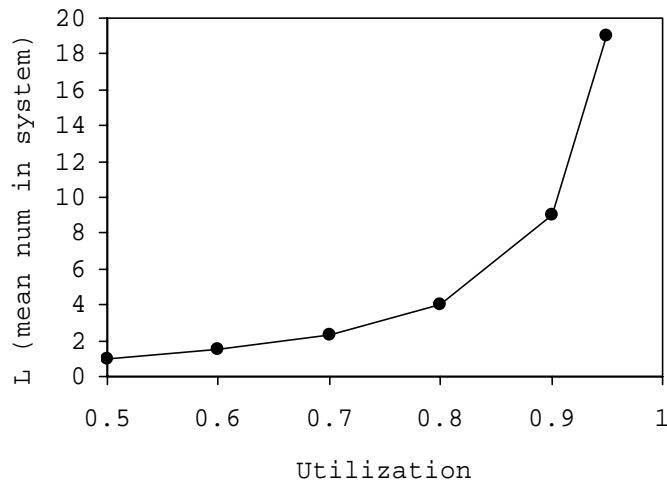
$Lq = L - \rho$ where ρ is the system utilization, which is arrival rate divided by service rate ($\rho = \lambda/\mu$).

$Wq = W - (1/\mu)$ where μ is the service rate.

Problem #6

For an M/M/1 queue, plot the mean number of customers in the system (L) for utilization of 0.50, 0.60, 0.70, 0.80, 0.90, and 0.95.

For an M/M/1 we know that $L = \frac{\rho}{1-\rho}$ so for the given values of utilization we compute L as 1.0, 1.5, 2.33, 4, 9, and 19. We can then plot:



Problem #7

Attached is `mm1_mini.c`. This is the M/M/1 simulation program from MacDougall and also as discussed in class. Modify the program such that a departing customer has a probability `PR_REDO` of immediately returning to the queue. The value of `PR_REDO` could be set as a constant (e.g., with a `#define`). Your modified `mm1_mini.c` models a single server queue with error checking where a departing customer with a detected error is immediately sent back for re-service. The probability of a departing customer having a detected error is `PR_REDO`.

See yellow highlighted code on the attached `mm1_mini.c` program.

Problem #8

List the basic components and give the flowchart of a discrete event simulation. Describe how an event list works.

The basic components are system state, simulation clock (time), event list, event routine, statistical counters, library routines, report generator, initialization routine, and main program. The flowchart is Figure 4.4 from Molloy (handed-out and/or discussed in class on Wednesday 02/09/05).

Extra Credit: (10 points)

Write a simulation program in C for problem #2. As with the previous problems, you may assume that `rand_val()` exists and can be called.

One possible solution is (`rand_val()` is not shown):

```
#include <stdio.h>

double rand_val(void);

void main(void)
{
    int total, i, s1, s2, s3, sum, count[4];

    total = 1000000;
    count[0] = count[1] = count[2] = count[3] = 0;
    for (i=0; i<total; i++)
    {
        sum = s1 = s2 = s3 = 0;
        if (rand_val() < 0.10) s1 = 1;
        if (rand_val() < 0.10) s2 = 1;
        if (rand_val() < 0.10) s3 = 1;

        sum = s1 + s2 + s3;

        count[sum]++;
    }

    printf("Pr[0 down] = %f \n", (double) count[0] / total);
    printf("Pr[1 down] = %f \n", (double) count[1] / total);
    printf("Pr[2 down] = %f \n", (double) count[2] / total);
    printf("Pr[3 down] = %f \n", (double) count[3] / total);
}
```

The output from this program is:

```
Pr[0 down] = 0.729539
Pr[1 down] = 0.242726
Pr[2 down] = 0.026753
Pr[3 down] = 0.000982
```

Which matches (as expected) the theoretical (analysis) results from problem #2.

```

//===== file = mml_mini.c =====
//= A simple M/M/1 queue simulation =
//=====
//= Notes: =
//= =
//= This program is adapted from "Figure 1.4" in Simulating Computer =
//= Systems, Techniques and Tools by M. H. MacDougall (1987) =
//=====
//= Build: gcc mml_mini -lm, bcc32 mml_mini.c, cl mml_mini.c =
//=====
//= Execute: mml_mini =
//=====
//= History: KJC (01/27/99) - Genesis =
//=====

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

//----- Constants -----
#define SIM_TIME 1.0e6 // Simulation time
#define PR_REDO 0.10 // Probability to redo a departing customer

//----- Function prototypes -----
double expntl(double x); // Generate exponential RV with mean x
double rand_val(void); // Generate unif(0,1) RV

//===== Main program =====
void main(void)
{
    double end_time = SIM_TIME; // Total time to simulate
    double Ta = 3.0; // Mean time between arrivals
    double Ts = 1.0; // 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)
    long int n = 0; // Number of customers in the system

    // 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
            if (rand_val() < PR_REDO) // Test if should redo or not
                t1 = time;
            else
                t1 = time + expntl(Ta);
            if (n == 1) // If first customer in system then
                t2 = time + expntl(Ts); // assign its departure time
        }
        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 + expntl(Ts); // assign next departure time
            else // If no customers in system then
                t2 = end_time; // assign next departure to "infinity"
        }
    }
}

```

```

//=====
//=  Function to generate exponentially distributed RVs           =
//=    - Input:  x (mean value of distribution)                 =
//=    - Output: Returns with exponential RV                    =
//=====
double expntl(double x)
{
    double z;          // Uniform random number from 0 to 1

    // Pull a uniform RV (0 < z < 1)
    do
    {
        z = rand_val();
    }
    while ((z == 0) || (z == 1));

    // Inversion formula for exponential RV
    return(-x * log(z));
}

//=====
//=  Multiplicative LCG for generating uniform(0.0, 1.0) random numbers =
//=    - x_n = 7^5*x_(n-1)mod(2^31 - 1)                         =
//=    - With x seeded to 1 the 10000th x value should be 1043618065 =
//=    - From R. Jain, "The Art of Computer Systems Performance Analysis," =
//=      John Wiley & Sons, 1991. (Page 443, Figure 26.2)     =
//=====
double rand_val(void)
{
    const long  a =      16807;  // Multiplier
    const long  m = 2147483647;  // Modulus
    const long  q =      127773;  // m div a
    const long  r =      2836;   // m mod a
    static long x =          1;   // Random int value (seed is set to 1)
    long        x_div_q;        // x divided by q
    long        x_mod_q;        // x modulo q
    long        x_new;          // New x value

    // RNG using integer arithmetic
    x_div_q = x / q;
    x_mod_q = x % q;
    x_new = (a * x_mod_q) - (r * x_div_q);
    if (x_new > 0)
        x = x_new;
    else
        x = x_new + m;

    // Return a random value between 0.0 and 1.0
    return((double) x / m);
}

```