

>>> SOLUTION <<<

Welcome to exam #2 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. 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, unless otherwise noted. This exam includes a copy of page 56 from MacDougall and Section 24 (Statements, Reserved Words) from the CSIM manual.

Good luck!!! ☺

Please answer:

I studied about _____ hours for this exam

I reviewed my exam #2 "cheat sheet" with Dr. Christensen - **Yes or No**

Outside of the class I spend about _____ hours per week on this class (reading, assignments, etc.)

Problem #1

A simulation model can be built using a specialized simulation language, a specialized application, graphical "point-and-shoot" package, high level language with function or class library, or just using a high level languages. Give examples and discuss the trade-offs of each of these methods.

GPSS and SLAM are specialized languages. Simulators are specialized applications – flight simulator and nuclear power plant simulator are two possible examples. OPNET is an example of a "point and shoot" package. SMPL and CSIM are examples of function libraries. C and Java are high level languages (HLL). Trade-offs include time to learn a new language (specialized languages require this), cost (point-and-shoot are expensive), specialization (for an application domain), and amount of code to be written or model development time (HLL force long model development times). In many ways, function library add-ons to HLLs are the best compromise.

Problem #2

Answer the following general questions about SMPL.

a) What is SMPL and what does it contain?

SMPL is a collection of C functions including:

- Random variate generation
- Event list management
- "Facility" constructs (e.g., queues)
- Collection of standard statistics
- Report generation

b) The SMPL view of systems consists of three constructs. Describe the three constructs of SMPL (facility is one, the other two you must know) and the general SMPL "view of the world".

The SMPL view of systems is based on facilities, tokens, and events. A facility is a work performing resource that is reserved and held. A token is an active entity moving through facilities. An event is a change in state of any systems entity.

c) Describe pre-emption in SMPL. That is, what does the `preempt ()` function do?

The `preempt()` function lets a high priority token get the facility by preempting (pushing out) any currently in service token. The pushed out lower priority token resumes service and finishes its remaining service time when all higher priority tokens have finished.

Problem #3

Attachment #1 is an SMPL program. Give the output from this program.

The output is:

```
TOP time = 1.00 event = 1
1 time = 1.00
1a time = 1.00 sched(2) = 11.00
TOP time = 5.00 event = 1
1 time = 5.00
TOP time = 11.00 event = 2
2 time = 11.00
TOP time = 11.00 event = 1
1 time = 11.00
1a time = 11.00 sched(2) = 31.00
TOP time = 31.00 event = 2
2 time = 31.00
```

```
**** Simulation Error at Time 31.000
      Empty Event List
```

Problem #4 (answer on the attachment)

Attachment #2 is an SMPL program that models an M/M/1 queue. Change this program to model an M/D/10/10 queue.

The changed and one new line of code are marked in yellow highlighter in attachment #2.

Problem #5

What is CSIM? Describe what it supports. Give its history. Explain what the CSIM `create()` function does. Explain how time passes, or increments, in a CSIM simulation.

CSIM is a C/C++ function library for process-oriented simulation. CSIM supports constructs for processes, facilities, storage, events, mailboxes, tables, processes classes, and random variate generation. CSIM was developed in 1984 by Herb Schwetman at MCC. CSIM allows functions to become independent running processes (sort of like threads) by using a `create()` function call typically invoked at the top of a function. That is, a `create()` changes a function to a process. Time passes only during `hold()` statements.

Problem #6

Attachment #3 is a CSIM program. Give the output from this program.

The output is:

```
BEGIN at 0.000000
(1) at 0.000000
(2) at 0.000000
(1) at 1.000000
(1) at 2.000000
(1) at 3.000000
(3) at 10.000000
(2) at 10.000000
(3) at 18.000000
(2) at 18.000000
(3) at 24.000000
(2) at 24.000000
(3) at 28.000000
END at 103.000000
```

Problem #7 (answer on the attachment)

Attachment #4 is a CSIM program (it is our well-known M/M/1 model). Modify it to measure and output the percentage of departing customers that have a delay greater than 10 milliseconds.

The changed and new code are marked in yellow highlighter in attachment #4.

Problem #8

Answer the following questions about generating random numbers:

a) What are the four desired properties of a random number generator?

- 1) Uniform and independent
- 2) Fast, simple, and low storage
- 3) Reproducible
- 4) Multiple streams

b) Describe three different ways to generate random numbers and explain for each of the four properties (from (a)) which is “yes” and which is “no”.

Physical process (1 = yes, 2 = no, 3 = no, 4 = yes)

Table (1 = yes, 2 = no, 3 = yes, 4 = yes)

Iterative formula or algorithm (1 = yes (if done right), 2 = yes, 3 = yes, 4 = yes)

c) What do we mean by “pseudo-random”?

A pseudo-random streams passes all statistical tests (i.e., uniform and independent) for randomness, but it not truly random because it is produced by a reproducible means (e.g., an algorithm or iterative formula).

d) What is a linear congruential generator (LCG)? What is the major issue with an LCG?

A LCG is an iterative formula of the form $z_{n+1} = (az_n + c) \text{ modulo } (m)$. The major issue with an LGC is its repeat or cycle length (getting stuck is another issue, but not the major issue). Careful selection of a , c , and m can make the cycle suitably large enough for simulation (as large as billions of values).

Extra Credit

Suggest possible improvements to CSIM.

Many good answers are possible. Well thought-out idea reflecting actual use of CSIM will get maximum points.

Attachment #1

```
//===== file = ex1.c =====
//= SMPL example =
//----- Include files -----
#include <stdio.h> // Needed for printf()
#include "smpl.h" // Needed for SMPL

void main(void)
{
    int customer; // Customer id
    int event; // Event (1 = request, 2 = release)
    int server; // Handle for server facility
    real x; // Temporary real value

    // Initialize SMPL subsystem
    smpl(0, "Test Program");

    // Initialize server facility (single server)
    server = facility("server", 1);

    // Schedule some requests for the facility
    schedule(1, 1.0, customer = 10);
    schedule(1, 5.0, customer = 20);

    // Loop forever
    while (1)
    {
        // "Cause" the next event on the event list
        cause(&event,&customer);
        printf("TOP time = %4.2f event = %d \n", time(), event);

        // Process the event
        switch(event)
        {
            case 1: // *** Request Server
                printf("1 time = %4.2f \n", time());
                if (request(server, customer, 0) == 0)
                {
                    schedule(2, x = (double) customer, customer);
                    printf("1a time = %4.2f sched(2) = %4.2f\n", time(), time() + x);
                }
                break;

            case 2: // *** Release server
                release(server, customer);
                printf("2 time = %4.2f \n", time());
                break;

            default: // *** If not case 1 or 2
                printf("DEFAULT - No more events... time to exit() \n");
                break;
        }
    }
}
```

Attachment #2

```
//===== file = mm1.c =====
//= A simple M/M/1 queue simulation using SMPL =
//= - Modified to M/D/10/10 for exam #2 solution =
//=====
//----- Include files -----
#include <stdio.h> // Needed for printf()
#include "smpl.h" // Needed for SMPL

//===== Main program =====
void main(void)
{
    real Ta = 200; // Mean interarrival time (seconds)
    real Ts = 100; // Mean service time
    real te = 1.0e6; // Total simulation time
    int customer = 1; // Customer id (always '1' for this simulation)
    int event; // Event (1 = arrival, 2 = request, 3 = completion)
    int server; // Handle for server facility

    // Initialize SMPL subsystem
    smpl(0, "M/D/10/10 Queue");

    // Initialize server facility (ten servers)
    server=facility("server", 10);

    // Schedule arrival event at time 0 to kick-off simulation
    schedule(1, 0.0, customer);

    // Loop while simulation time is less than te
    while (time() < te)
    {
        // "Cause" the next event on the event list
        cause(&event,&customer);

        // Process the event
        switch(event)
        {
            case 1: // *** Arrival
                if (status(server) == 0)
                    schedule(2, 0.0, customer);
                schedule(1, expntl(Ta), customer);
                break;

            case 2: // *** Request Server
                if (request(server, customer, 0) == 0)
                    schedule(3, Ts, customer);
                break;

            case 3: // *** Release server
                release(server, customer);
                break;
        }
    }

    // Output standard SMPL report
    report();
}
```

Attachment #3

```
//===== file = ex2.c =====
// CSIM example =
//=====
//----- Include files -----
#include <stdio.h>      // Needed for printf()
#include "csim.h"      // Needed for CSIM stuff

//----- Function prototypes -----
void procl(double hold_time); // Process #1

//----- Globals -----
FACILITY Server;      // Facility for a server

//===== Main program =====
void sim(void)
{
    create("sim");

    Server = facility("Server facility");

    printf("BEGIN at %f \n", clock);
    procl(10.0);
    hold(1.0);
    procl(8.0);
    hold(1.0);
    procl(6.0);
    hold(1.0);
    procl(4.0);
    hold(100.0);
    printf("END at %f \n", clock);
}

void procl(double hold_time)
{
    create("process #1");

    printf("(1) at %f \n", clock);
    reserve(Server);
    printf("(2) at %f \n", clock);
    hold(hold_time);
    release(Server);
    printf("(3) at %f \n", clock);
}
```

Attachment #4

```
//===== file = mml_csim.c =====
//= A CSIM simulation of an M/M/1 queueing system =
//=====
#include <stdio.h> // Needed for printf()
#include "csim.h" // Needed for CSIM18 stuff

FACILITY Server; // Declaration of CSIM Server facility
int Over_10; // Counter for over 10 millisecc
int Depart; // Departure counter

void generate(double lambda, double mu); // Customer generator
void queuel(double org_time, double service_time);

void sim(void)
{
    double lambda; // Mean arrival rate (cust/sec)
    double mu; // Mean service rate (cust/sec)

    create("sim");

    Server = facility("Server");

    Depart = Over_10 = 0;
    lambda = 100.0; mu = 300.0;

    generate(lambda, mu);
    hold(100000.0);

    printf("===== \n");
    printf("= Mean num in system = %6.3f cust \n", qlen(Server));
    printf("= Mean response time = %6.3f sec \n", resp(Server));
    printf("= Over 10 ms = %f %% \n",
    100.0 * ((double) Over_10 / Depart));
    printf("===== \n");
}

void generate(double lambda, double mu)
{
    double interarrival_time; // Interarrival time to next send
    double service_time; // Service time for this customer

    create("generate");

    while(1)
    {
        interarrival_time = exponential(1.0 / lambda);
        hold(interarrival_time);

        service_time = exponential(1.0 / mu);
        queuel(clock, service_time);
    }
}

void queuel(double org_time, double service_time)
{
    create("queuel");

    reserve(Server);
    hold(service_time);
    release(Server);

    Depart++;
    if ((clock - org_time) > 0.010)
        Over_10++;
}
}
```