>>> Assignment #5 for Simulation (CAP 4800) <<<

>>> SOLUTIONS <<<

This assignment covers material related to beginning CSIM topics. These problems also help establish “building blocks” for your semester project.

Problem #1 (50 points)

Modify the existing CSIM M/M/1 simulation (`mm1_csim.c`) to read interarrival times from an input file. Let the mean service time be fixed to 1.0 seconds per customer. Create a file with 1 million exponentially distributed service time values such that the offered load to the queue is 80%. Run the simulation and submit a screenshot of the results. Submit also a listing of the key function (the `generate()`function) that you modified for this problem. **Hint:** You can use `genexp.c` (found on Christensen tools page) to generate exponentially distributed random variables.

Note that you will need to change the stopping criterion for the simulation. To do this make the following changes.

1) Add `EVENT DoneEvent;` to globals
2) Add `DoneEvent = event(“Done event”);` to CSIM initializations in `main()`
3) Change `hold(SIM_TIME);` to `wait(DoneEvent);` in `main()`
4) When you have read all the values from the input file (that is, you are at end-of-file condition) you will need to execute the line of code `set(DoneEvent);` to terminate the simulation

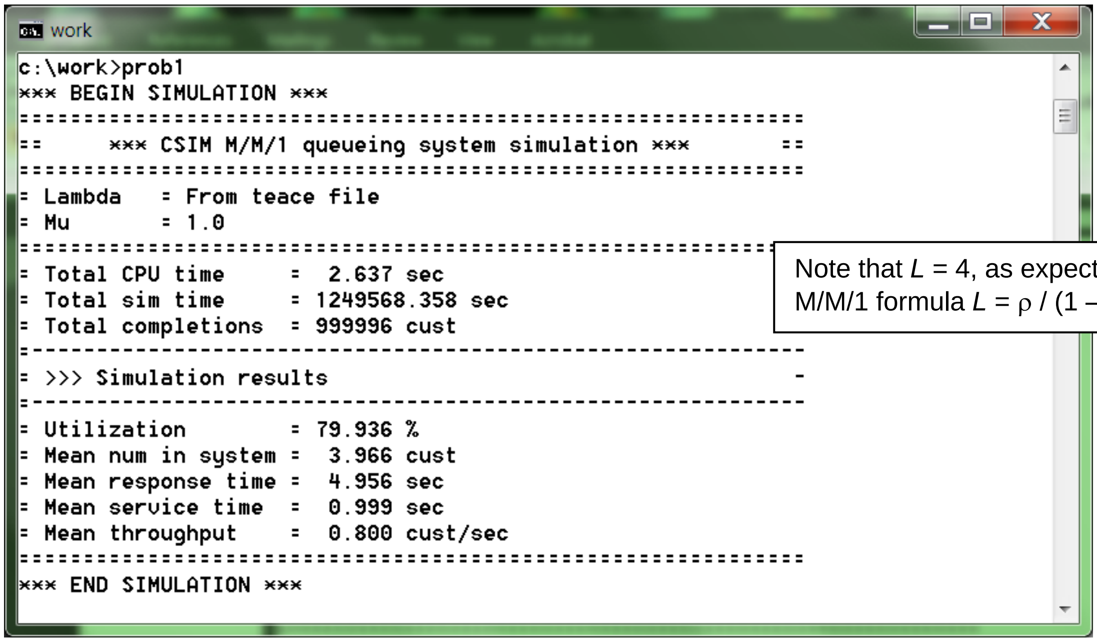
You may also need to make minor changes to `main()` (especially to the output section) since lambda is no longer a parameter set in `main()`.

Here is the modified generate() function:

```
//==============================================================================
//==  Function to generate Poisson customers                                  ==
//==   - Reads interarrival times from "trace.txt"                            ==
//==============================================================================
void generate()
{
  FILE     *fp;                  // File pointer
  char     inString[256];        // Input string from file
  double   interarrival_time;    // Interarrival time to next send
  double   service_time;         // Service time for this customer

  create("generate");

  // Open the trace file
  fp = fopen("trace.txt", "r");
  if (fp == NULL)
  {
    printf("*** ERROR - file does not exist \n");
    exit(1);
  }

  // Loop until eof to generate customers
  while(1)
  {
    // Pull an interarrival time from input file and hold for it
    fscanf(fp,"%s", inString);
    if (feof(fp)) break;
    interarrival_time = atof(inString);
    hold(interarrival_time);
```

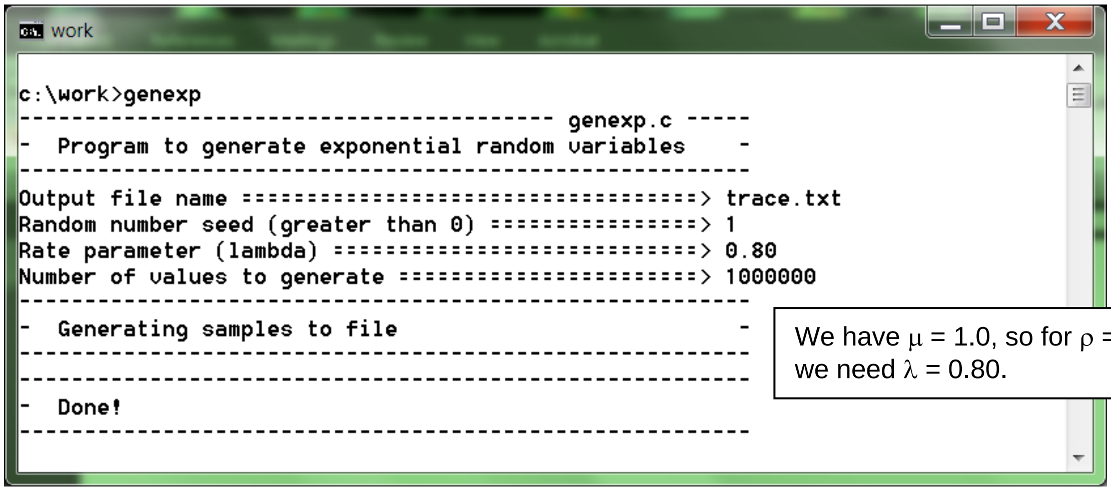
```
      // Pull a service time with mean 1.0 and then send the customer to the queue
      service_time = exponential(1.0);
      queue1(service_time);
    }

    // Terminate the simulation
    set(DoneEvent);
  }
```

And, here is a screen shot of an execution (also shown is the genexp.c execution to generate the values in trace.txt):



```
c:\work>prob1
*** BEGIN SIMULATION ***
==============================================================
==     *** CSIM M/M/1 queueing system simulation ***       ==
==============================================================
= Lambda    = From teace file
= Mu        = 1.0
==============================================================
= Total CPU time     =  2.637 sec
= Total sim time     = 1249568.358 sec
= Total completions  = 999996 cust
=-------------------------------------------------------------
= >>> Simulation results                                     -
=-------------------------------------------------------------
= Utilization        = 79.936 %
= Mean num in system =  3.966 cust
= Mean response time =  4.956 sec
= Mean service time  =  0.999 sec
= Mean throughput    =  0.800 cust/sec
==============================================================
*** END SIMULATION ***
```

Note that *L* = 4, as expect from our M/M/1 formula *L* = ρ / (1 – ρ).



```
c:\work>genexp
------------------------------------------ genexp.c -----
-  Program to generate exponential random variables    -
---------------------------------------------------------
Output file name ===================================> trace.txt
Random number seed (greater than 0) ================> 1
Rate parameter (lambda) ============================> 0.80
Number of values to generate ======================> 1000000
---------------------------------------------------------
-  Generating samples to file                           -
---------------------------------------------------------
---------------------------------------------------------
-  Done!
---------------------------------------------------------
```

We have μ = 1.0, so for ρ = 0.80 we need λ = 0.80.

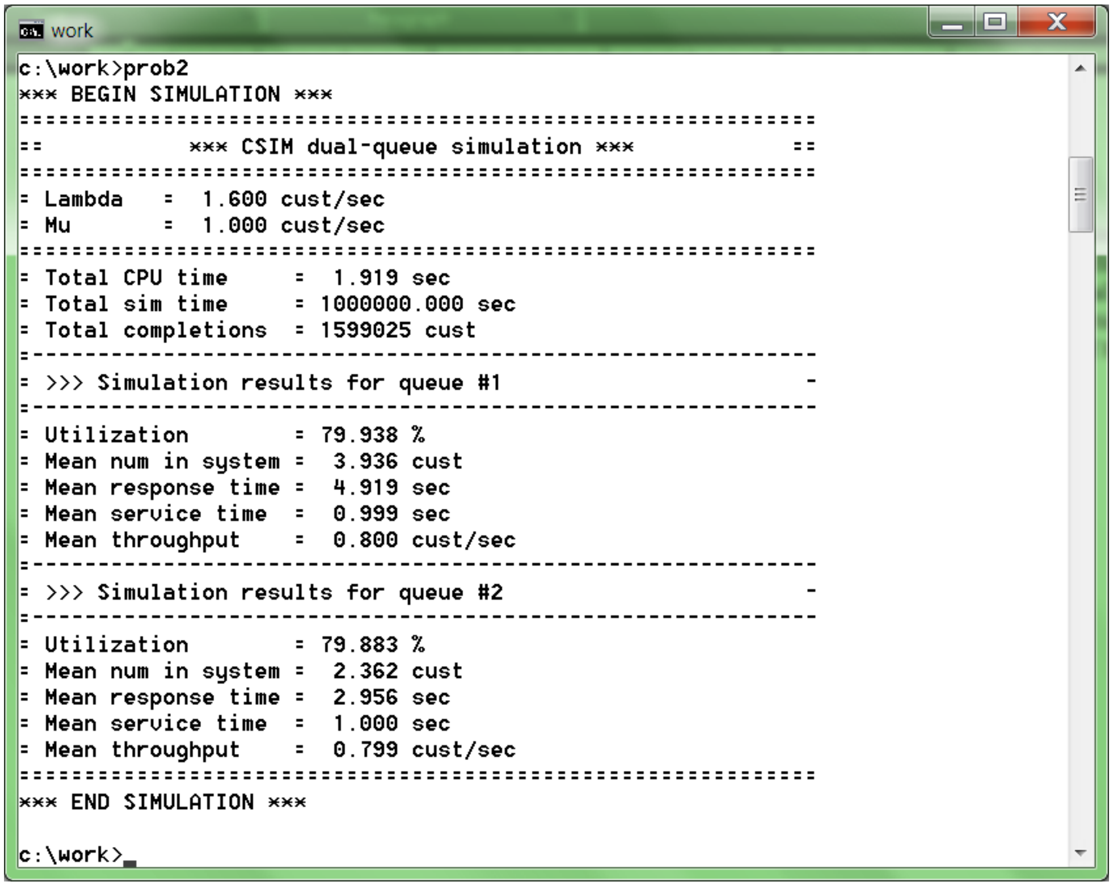
Problem #2 (50 points)

Modify the existing CSIM M/M/1 simulation (`mm1_csim.c`) to have a second queue that has deterministic service time and change the `generate()`function to probabilistically split (with 50/50 probability – a coin flip) the generated customers to each queue. That is, half of the generated customers should go to the queue with exponential service time (the queue that already exists in the model) and the other half to the new queue with deterministic service time. Fix the mean service time for each queue to be 1.0 seconds per customer. Run the simulation such that each queue has an offered load of 80% and submit a screen shot of the results. Run the simulation for a simulated 1 million seconds.

In addition to changing the `generate()` function you will need to do the following:

1) Add another queue() function
2) Add and initialize another Server FACILITY
3) Carefully update the output results section of `main()`
4) Carefully update the header block and inline comments

Here is a screen shot of an execution. Starting on the next page is the source code.



```
c:\work>prob2
*** BEGIN SIMULATION ***
============================================================
==           *** CSIM dual-queue simulation ***           ==
============================================================
= Lambda   =  1.600 cust/sec
= Mu       =  1.000 cust/sec
============================================================
= Total CPU time     =  1.919 sec
= Total sim time     = 1000000.000 sec
= Total completions  = 1599025 cust
=-----------------------------------------------------------
= >>> Simulation results for queue #1                      -
=-----------------------------------------------------------
= Utilization        = 79.938 %
= Mean num in system =  3.936 cust
= Mean response time =  4.919 sec
= Mean service time  =  0.999 sec
= Mean throughput    =  0.800 cust/sec
=-----------------------------------------------------------
= >>> Simulation results for queue #2                      -
=-----------------------------------------------------------
= Utilization        = 79.883 %
= Mean num in system =  2.362 cust
= Mean response time =  2.956 sec
= Mean service time  =  1.000 sec
= Mean throughput    =  0.799 cust/sec
============================================================
*** END SIMULATION ***

c:\work>
```

```
//====================================================== file = prob2.c =====
//=  A CSIM simulation of a dual-queue system                               =
//=   - For problem #2 of assignment #5 for Simulation for summer 2013        =
//===========================================================================
//=  Notes: 1) Values for lambda and mu are set in the main program          =
//=-------------------------------------------------------------------------=
//= Example execution:                                                      =
//=                                                                         =
//=   *** BEGIN SIMULATION ***                                              =
//=   =============================================================         =
//=   ==           *** CSIM dual-queue simulation ***            ==         =
//=   =============================================================         =
//=   = Lambda   =  1.600 cust/sec                                          =
//=   = Mu       =  1.000 cust/sec                                          =
//=   =============================================================         =
//=   = Total CPU time     =  1.857 sec                                     =
//=   = Total sim time     = 1000000.000 sec                                =
//=   = Total completions  = 1599025 cust                                   =
//=   =------------------------------------------------------------         =
//=   = >>> Simulation results for queue #1                       -         =
//=   =------------------------------------------------------------         =
//=   = Utilization        = 79.938 %                                       =
//=   = Mean num in system =  3.936 cust                                    =
//=   = Mean response time =  4.919 sec                                     =
//=   = Mean service time  =  0.999 sec                                     =
//=   = Mean throughput    =  0.800 cust/sec                                =
//=   =------------------------------------------------------------         =
//=   = >>> Simulation results for queue #2                       -         =
//=   =------------------------------------------------------------         =
//=   = Utilization        = 79.883 %                                       =
//=   = Mean num in system =  2.362 cust                                    =
//=   = Mean response time =  2.956 sec                                     =
//=   = Mean service time  =  1.000 sec                                     =
//=   = Mean throughput    =  0.799 cust/sec                                =
//=   =============================================================         =
//=   *** END SIMULATION ***                                                =
//=-------------------------------------------------------------------------=
//=  Build: standard CSIM build                                             =
//=-------------------------------------------------------------------------=
//=  Execute: prob2                                                         =
//=-------------------------------------------------------------------------=
//=  Author: Ken Christensen                                                =
//=          University of South Florida                                    =
//=          WWW: http://www.csee.usf.edu/~christen                         =
//=          Email: christen@csee.usf.edu                                   =
//=-------------------------------------------------------------------------=
//=  History: KJC (06/19/11) - Genesis (from mm1_csim.c)                    =
//===========================================================================
//----- Includes -------------------------------------------------------------
#include <stdio.h>      // Needed for printf()
#include "csim.h"       // Needed for CSIM stuff

//----- Defines --------------------------------------------------------------
#define SIM_TIME 1.0e6  // Total simulation time in seconds
```

```
//----- Globals -----------------------------------------------------------
FACILITY Server1;       // Declaration of CSIM Server facility #1
FACILITY Server2;       // Declaration of CSIM Server facility #2

//----- Prototypes --------------------------------------------------------
void generate(double lambda, double mu);  // Customer generator
void queue1(double service_time);         // Single server queue #1
void queue2(double service_time);         // Single server queue #2

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

  // Create the simulation
  create("sim");

  // CSIM initializations
  Server1 = facility("Server #1");
  Server2 = facility("Server #2");

  // Parameter initializations
  lambda = 1.6;
  mu = 1.0;

  // Output begin-of-simulation banner
  printf("*** BEGIN SIMULATION *** \n");

  // Initiate generate function and hold for SIM_TIME
  generate(lambda, mu);
  hold(SIM_TIME);

  // Output results
  printf("============================================================= \n");
  printf("==           *** CSIM dual-queue simulation ***            == \n");
  printf("============================================================= \n");
  printf("= Lambda   = %6.3f cust/sec   \n", lambda);
  printf("= Mu       = %6.3f cust/sec   \n", mu);
  printf("============================================================= \n");
  printf("= Total CPU time     = %6.3f sec      \n", cputime());
  printf("= Total sim time     = %6.3f sec      \n", clock);
  printf("= Total completions  = %ld cust       \n",
    completions(Server1) + completions(Server2));
  printf("=------------------------------------------------------------ \n");
  printf("= >>> Simulation results for queue #1                       - \n");
  printf("=------------------------------------------------------------ \n");
  printf("= Utilization        = %6.3f %%       \n", 100.0 * util(Server1));
  printf("= Mean num in system = %6.3f cust     \n", qlen(Server1));
  printf("= Mean response time = %6.3f sec      \n", resp(Server1));
  printf("= Mean service time  = %6.3f sec      \n", serv(Server1));
  printf("= Mean throughput    = %6.3f cust/sec \n", tput(Server1));
  printf("=------------------------------------------------------------ \n");
  printf("= >>> Simulation results for queue #2                       - \n");
  printf("=-------------------------------------------------------------- \n");
  printf("= Utilization        = %6.3f %%       \n", 100.0 * util(Server2));
  printf("= Mean num in system = %6.3f cust     \n", qlen(Server2));
  printf("= Mean response time = %6.3f sec      \n", resp(Server2));
  printf("= Mean service time  = %6.3f sec      \n", serv(Server2));
  printf("= Mean throughput    = %6.3f cust/sec \n", tput(Server2));
  printf("============================================================== \n");

  // Output end-of-simulation banner
  printf("*** END SIMULATION *** \n");
}

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

  create("generate");

  // Loop forever to create customers
  while(1)
  {
    // Pull an interarrival time and hold for it
    interarrival_time = exponential(1.0 / lambda);
    hold(interarrival_time);

    // Probabilistic split to queue1() and queue2()
    if (uniform(0.0, 1.0) <= 0.50)
    {
      service_time = exponential(1.0 / mu);  // Exponential service time
      queue1(service_time);
    }
    else
    {
      service_time = 1.0 / mu;               // Determinisitic service time
      queue2(service_time);
    }
  }
}

//=============================================================================
//==  Function for single server queue #1                                    ==
//=============================================================================
void queue1(double service_time)
{
  create("queue1");

  // Reserve, hold, and release server
  reserve(Server1);
  hold(service_time);
  release(Server1);
}
```

```
//==============================================================================
//==  Function for single server queue #2                                    ==
//==============================================================================
void queue2(double service_time)
{
  create("queue2");

  // Reserve, hold, and release server
  reserve(Server2);
  hold(service_time);
  release(Server2);
}
```