Solution to Homework #1

Problem 1:

Output:

```
thebe% gcc -g3 -I. hw01.c smpl.c rand.c bmeans.c stat.c -lm -o hw01
thebe% hw01
batch 1 mean = 0.725
batch 2 mean = 0.729
batch 3 mean = 0.703
batch 4 mean = 0.748
batch 5 mean = 0.725
batch 6 mean = 0.710
batch 7 mean = 0.651
batch 8 mean = 0.717
batch 9 mean = 0.727
mean is 0.714871 and half width is 0.018836 after 10 batches
(1) customer turned away probability is 0.277794
(2) Average number of customers waiting in the system is 4.776910
(3) Response time per client for faster server is 0.646413
(4) Response time per client for slower server is 0.771224
(5) Throughput is 10.793293
(6) Confidence accuracy is 0.026349
```

Source Code:

```
/**********************************************************************/
/* M/M/3/10 queue */
/**********************************************************************/
#include "smpl.h"
#define TOKENS 1000
#define TRUE 1
#define FALSE 0
int main()
{
    real Ta=1.0/15,Ts_f=1.0/5,Ts_s1=1.0/3,Ts_s2=1.0/3,mean,hw;
    int tk_id=0,customer=0,event,server,nb;
    real ts[TOKENS]; /* start time stamp */
    int cont=TRUE;
    smpl(0,"M/M/3/10 Queue");
    init_bm(200000);
    server=facility("server",3);
    int flag_f = 0; int flag_s1 = 0; int flag_s2 = 0;
    real count_arr = 0; real count_turn = 0;
    int Cf = 0; int Cs1 = 0; int Cs2 = 0;
    real Wf = 0; real Ws1 = 0; real Ws2 = 0;
    schedule(1,0.0,tk_id);
    while (cont)
    {
        cause(&event,&customer);
        switch(event)
        {
        case 1: /* arrival */
            ++count_arr;
            ts[customer] = time();
            if(inq(server)<7)schedule(2,0.0,customer);
            else
                ++count_turn;
            if (++tk_id >= TOKENS) tk_id=0;
            schedule(1,expntl(Ta),tk_id);
```
break;
case 2: /* request server */
   if (request(server,customer,0)==0)
   |
   if(flag_f==0){//fast
       schedule(4,expntl(Ts_f),customer);
       flag_f=1;
   }
   else if(flag_s1==0)
       schedule(3,expntl(Ts_s1),customer);
   flag_s1=1;
   else if(flag_s2==0)
       schedule(3,expntl(Ts_s2),customer);
   flag_s2=1;
   |
   break;
case 3: /* release slow server */
   release(server,customer);
   if(flag_s1==1){
       flag_s1=0;
       Cs1++;
       Ws1 =Ws1+(time()-ts[customer]);
   }
   if(flag_s2==1){
       flag_s2=0;
       Cs2++;
       Ws2 =Ws2+(time()-ts[customer]);
   }
   if (obs(time()-ts[customer]) == 1) cont = FALSE;
   break;
case 4: /* release fast server */
   release(server,customer);
   flag_f=0;
   Cf++;
   Wf = Wf + (time()-ts[customer]);
   if (obs(time()-ts[customer]) == 1) cont = FALSE;
   break;
|
civals(&mean, &hw, &nb);
printf("Mean is %f and half width is %f after %d batches\n", mean, hw, nb);
printf("[1] Customer turned away probability is %f \n", count_turn/count_arr);
printf("[2] Average number of customers waiting in the system is %f \n", Lq(server));
printf("[3] Response time per client for faster server is %f \n", Wf/Cf);
printf("[4] Response time per client for slower server is %f \n", (Ws1+Ws2)/(Cs1+Cs2));
printf("[5] Throughput is %f \n", (Cf+Cs1+Cs2)/time());
printf("[6] Confidence accuracy is %f \n", hw/mean);
return 0;
 |
|
|
|
|
|
|
|

Problem 2:

a. Write a Sharpe code based on a reliability graph model to compute the system reliability after 3 weeks of operation.
2. Source code:

bind
b. Find the minimal path and minimal cut sets.

the minimal paths are: {1,4,5,7},{1,4,6},{1,2,5,6},{1,2,7},{3,7},{3,5,6},{3,2,4,6}

the minimal cuts are: {1,3},{1,2,5,7},{6,7},{2,3,4},{4,5,7},{2,3,5,6}

c. Build a fault tree model based on the minimal path set identified in (b) and then write a Sharpe code based on your fault tree model to compute the system reliability after 3 weeks of operation. The answer obtained here should be the same as that obtained from part (a).

1. fault tree model

![Fault Tree Model](image)

2. Output : 1-value(504;system1): 9.9907e-01

3. Source code:

```sharpe
bind
mA 0.00007
mB 0.00006
mC 0.00005
expr 1-value(504;system1)
End
```
mD 0.00004
mE 0.00003
mF 0.00002
mG 0.00001
end
ftree system1
repeat A exp(mA)
repeat B exp(mB)
repeat C exp(mC)
repeat D exp(mD)
repeat E exp(mE)
repeat F exp(mF)
repeat G exp(mG)
or ADF A D F
or ADEG A D E G
or ABKF A B K F
or ABG A B G
or CEF C E F
or CG C G
or CBDF C B D F
and top ADF ADEG ABKF ABG CEF CG CBDF
end
* printing: 5 decimal places
*format 5
*cdf
*cdf(block1a)
expr l-value(504;system1)
end
end