Problem 1
The SPN model for M/M/3/10 is:

![SPN model diagram]

Code for calculating turn-away probability, population, throughput and response time:

```c
#include"user.h"

double lambda=5.0;

double mu;

int b=10;

int m=3;

double ratio;

parameters()
{
    ratio = input("enter lambda over mu ratio");
    mu = lambda/ratio;
}

rate_type rate_serv()
{
    if (mark("buf")<m) return (mark("buf")*mu);
    else return (m*mu);
}

net()
{
    place("buf");
    trans("trin");
    trans("trserv");
    rateval("trin", lambda);
    ratefun("trserv", rate_serv);
}```
oarc("trin","buf");
iarc("trserv","buf");
mharc("trin","buf",b);
}

assert()
{
    if (mark("buf") > b) return(RES_ERROR);
    else return(RES_NOERR);
}
ac_init(){pr_net_info();}
ac_reach(){pr_rg_info();}

/*reward assignment functions for calculating performance metrics*/
reward_type probrej() { if (mark("buf")==b) return (1.0); else return (0.0); }
reward_type n(){return(mark("buf"));}
reward_type X() {return(rate("trserv"));}

ac_final() {
    printf("turned-away probability = %f\n", expected(probrej));
    printf("population = %f\n", expected(n));
    printf("throughput = %f\n", expected(X));
    printf("response time = %f\n", expected(n)/expected(X));
}

Output:

SPNP Version 4.0
The analysis is starting.
INPUT "enter lambda over mu ratio">
1
The reachability graph contains:
   11 tangible markings
   0 vanishing markings
   20 arcs
After elimination of redundant arcs:
   # of remaining arcs: 20
After the elimination of vanishing markings:
   # of remaining arcs: 20
Solving the Markov chain...
...Markov chain solved
Reading the reachability graph info ...

**turned-away probability = 0.000028**

**population = 1.045310**

**throughput = 4.999864**

**response time = 0.209068**

End of execution.

janus% !!
spnp hw3-1
SPNP Version 4.0
The analysis is starting.
**INPUT "enter lambda over mu ratio" -> 2.5**
The reachability graph contains:
   11 tangible markings
   0 vanishing markings
   20 arcs
After elimination of redundant arcs:
   # of remaining arcs: 20
After the elimination of vanishing markings:
   # of remaining arcs: 20
Solving the Markov chain...
...Markov chain solved
Reading the reachability graph info ...

**turned-away probability = 0.039040**

**population = 4.061432**

**throughput = 4.804803**

**response time = 0.845286**

End of execution.

janus% !!
spnp hw3-1
SPNP Version 4.0
The analysis is starting.
**INPUT "enter lambda over mu ratio" -> 5**
The reachability graph contains:
   11 tangible markings
0 vanishing markings
20 arcs
After elimination of redundant arcs:
  # of remaining arcs: 20
After the elimination of vanishing markings:
  # of remaining arcs: 20
Solving the Markov chain...
  ...Markov chain solved
Reading the reachability graph info ...
  *turned-away probability = 0.402758*
  population = 8.566413
  throughput = 2.986198
  response time = 2.868668
End of execution.

janus% !
spnp hw3-1
SPNP Version 4.0
The analysis is starting.
INPUT "enter lambda over mu ratio">
7.5
The reachability graph contains:
  11 tangible markings
  0 vanishing markings
  20 arcs
After elimination of redundant arcs:
  # of remaining arcs: 20
After the elimination of vanishing markings:
  # of remaining arcs: 20
Solving the Markov chain...
  ...Markov chain solved
Reading the reachability graph info ...
  *turned-away probability = 0.600085*
  population = 9.334678
  throughput = 1.999570
  response time = 4.668343
End of execution.

janus% !!
The analysis is starting.

INPUT "enter lambda over mu ratio"> 10

The reachability graph contains:
   11 tangible markings
   0 vanishing markings
   20 arcs

After elimination of redundant arcs:
   # of remaining arcs: 20

After the elimination of vanishing markings:
   # of remaining arcs: 20

Solving the Markov chain...
...Markov chain solved

Reading the reachability graph info ...

turned-away probability = 0.700006
population = 9.571513
throughput = 1.499966
response time = 6.381152

End of execution.

Code for calculating Customers served over [0, 7200] (A separate program is needed for transient analysis):

```c
#include"user.h"
double lambda=5.0;
double mu;
int b=10;
int m=3;
double ratio;

parameters()
{
    ratio = input("enter lambda over mu ratio");
    mu = lambda/ratio;
iopt(IOP_METHOD,VAL_TSUNIF);
}

rate_type rate_serv()
```
if (mark("buf")<m) return (mark("buf")*mu);
else return (m*mu);
}

net()
{
place("buf");
trans("trin");
trans("trserv");
rateval("trin", lambda);
ratefun("trserv", rate_serv);
oarc("trin","buf");
iarc("trserv","buf");
mharc("trin","buf",b);
}

assert()
{
  if (mark("buf") > b) return(RES_ERROR);
  else return(RES_NOERR);
}
ac_init(){pr_net_info();}
ac_reach(){pr_rg_info();}

/*reward assignment functions for calculating performance metrics*/
reward_type X() {return(rate("trserv"));}

ac_final() {
time_value(7200.0);
printf("Customers served over [0, 7200] = %f\n", cum_expected(X));
}

Output:

janus% !!
spnp hw3-1ee
SPNP Version 4.0
The analysis is starting.
INPUT "enter lambda over mu ratio”>
1
The reachability graph contains:
11 tangible markings
0 vanishing markings
20 arcs
After elimination of redundant arcs:
  # of remaining arcs: 20
After the elimination of vanishing markings:
  # of remaining arcs: 20
Solving the Markov chain...
SS reached before LTP in Unif. at 120
Cumulative Uniformization: RTP = 153558
SS reached before RTP in Cum. Unif. at 170
...Markov chain solved
Reading the reachability graph info ...

**Customers served over [0, 7200] = 35997.957181**

End of execution.

janus% !!
spnp hw3-1ee
SPNP Version 4.0

The analysis is starting.
INPUT "enter lambda over mu ratio" >
2.5
The reachability graph contains:
11 tangible markings
0 vanishing markings
20 arcs
After elimination of redundant arcs:
  # of remaining arcs: 20
After the elimination of vanishing markings:
  # of remaining arcs: 20
Solving the Markov chain...
SS reached before LTP in Unif. at 310
Cumulative Uniformization: RTP = 84977
SS reached before RTP in Cum. Unif. at 480
...Markov chain solved
Reading the reachability graph info ...

**Customers served over [0, 7200] = 34591.098382**

End of execution.

janus% !!
spnp hw3-1ee
SPNP Version 4.0
The analysis is starting.
INPUT "enter lambda over mu ratio">
5
The reachability graph contains:
   11 tangible markings
   0 vanishing markings
   20 arcs
After elimination of redundant arcs:
   # of remaining arcs: 20
After the elimination of vanishing markings:
   # of remaining arcs: 20
Solving the Markov chain...
SS reached before LTP in Unif. at 250
Cumulative Uniformization: RTP = 62049
SS reached before RTP in Cum. Unif. at 360
...Markov chain solved
Reading the reachability graph info ...

**Customers served over [0, 7200] = 21498.610526**
End of execution.

janus% ! !
spnp hw3-1ee
SPNP Version 4.0
The analysis is starting.
INPUT "enter lambda over mu ratio">
7.5
The reachability graph contains:
   11 tangible markings
   0 vanishing markings
   20 arcs
After elimination of redundant arcs:
   # of remaining arcs: 20
After the elimination of vanishing markings:
   # of remaining arcs: 20
Solving the Markov chain...
SS reached before LTP in Unif. at 160
Cumulative Uniformization: RTP = 54377
SS reached before RTP in Cum. Unif. at 230
...Markov chain solved
Reading the reachability graph info ...

**Customers served over [0, 7200] = 14395.822275**
End of execution.
Problem 2

```c
/* HW3, Problem 2.a */
#include <stdio.h>
#include <math.h>
#include "user.h"

double
 cpufailrate=1.0/7200.0,memfailrate=1.0/14400.0,cpureprate=1.0/240.0,memreprate=1.0/120
```

```
Janus% !!
spnp hw3-1ee
SPNP Version 4.0
The analysis is starting.
INPUT "enter lambda over mu ratio">
10
The reachability graph contains:
  11 tangible markings
  0 vanishing markings
  20 arcs
After elimination of redundant arcs:
  # of remaining arcs: 20
After the elimination of vanishing markings:
  # of remaining arcs: 20
Solving the Markov chain...
SS reached before LTP in Unif. at 130
Cumulative Uniformization: RTP = 50579
SS reached before RTP in Cum. Unif. at 170
...Markov chain solved
Reading the reachability graph info ...

Customers served over [0, 7200] = 10799.020209
End of execution.
```
parameters()
{
opt(IOP_METHOD,VAL_TSUNIF);
}

enabling_type cpuRepairEnbl()
{
    if (mark("cpu_up") >= 1)
        return 1;
    else return 0;
}

enabling_type memRepairEnbl()
{
    if (mark("mem_up") >= 1)
        return 1;
    else return 0;
}

rate_type cpu_repair()
{
    return(mark("cpu_down")*cpureprate);
}

rate_type mem_repair()
{
    return(mark("mem_down")*memreprate);
}

rate_type cpu_failure()
{
    return(mark("cpu_up")*cpufailrate);
}

rate_type mem_failure()
{
    return(mark("mem_up")*memfailrate);
}

net()
{
    place("cpu_up"); init("cpu_up",3);
    place("cpu_down");
    place("mem_up"); init("mem_up",2);
    place("mem_down");

    trans("cpu_failure");
    trans("cpu_repair");
    trans("mem_failure");
    trans("mem_repair");

    ratefun("cpu_failure", cpu_failure);
    ratefun("cpu_repair", cpu_repair);
    ratefun("mem_failure", mem_failure);
    ratefun("mem_repair", mem_repair);

    iarc("cpu_failure","cpu_up");
    iarc("cpu_repair","cpu_down");

    iarc("mem_failure","mem_up");
    iarc("mem_repair","mem_down");
```c
#include <stdio.h>
#include <math.h>
#include "user.h"

double cpufailrate=1.0/7200.0, memfailrate=1.0/14400.0, cpureprate=1.0/240.0, memreprate=1.0/120.0;

parameters()
{
    iopt(IOP_METHOD, VAL_TSUNIF);
}
```

Output:

SPNP Version 4.0
The analysis is starting.
The reachability graph contains:
    12 tangible markings (1 absorbing)
    0 vanishing markings
    27 arcs
After elimination of redundant arcs:
    # of remaining arcs: 27
ERROR/WARNING: transient initial marking. 5 markings reach it.
After the elimination of vanishing markings:
    # of remaining arcs: 27
Solving the Markov chain...
Uniformization: LTP = 145 and RTP = 290
Cumulative Uniformization: RTP = 313
...Markov chain solved
Reading the reachability graph info ...  
**Reliability at time 12000 = 0.981937**  
End of execution.
enabling_type cpuRepairEnbl()
{
    if (mark("cpu_up") >= 1)
        return 1;
    else return 0;
}

enabling_type memRepairEnbl()
{
    if (mark("mem_up") >= 1)
        return 1;
    else return 0;
}

double rate_type cpu_repair()
{
    return cpureprate;
}

double rate_type mem_repair()
{
    return memreprate;
}

double rate_type cpu_failure()
{
    return mark("cpu_up")*cpufailrate;
}

double rate_type mem_failure()
{
    return mark("mem_up")*memfailrate;
}

net()
{
    place("cpu_up"); init("cpu_up",3);
    place("cpu_down");
    place("mem_up"); init("mem_up",2);
    place("mem_down");

    trans("cpu_failure");
    trans("cpu_repair");
    trans("mem_failure");
    trans("mem_repair");
    ratefun("cpu_failure", cpu_failure);
    ratefun("cpu_repair", cpu_repair);
    ratefun("mem_failure", mem_failure);
    ratefun("mem_repair", mem_repair);
    iarc("cpu_failure","cpu_up");
    iarc("cpu_repair","cpu_down");
    iarc("mem_failure","mem_up");
    iarc("mem_repair","mem_down");
    oarc("cpu_failure","cpu_down");
    oarc("cpu_repair","cpu_up");
    oarc("mem_failure","mem_down");
    oarc("mem_repair","mem_up");
enabling("cpu_repair", cpuRepairEnbl);
enabling("mem_repair", memRepairEnbl);
}

assert();
ac_init() {pr_net_info();}
ac_reach() {pr_rg_info();}

reward_type reliability(){
if (mark("cpu_up") >= 1 && mark("mem_up") >= 1) return(1.0);
else return(0.0);
}

ac_final(){
time_value(12000.0);
printf("Reliability at time 12000 = %f\n", expected(reliability));
}

Output:
SPNP Version 4.0
The analysis is starting.
The reachability graph contains:
   12 tangible markings (1 absorbing)
   0 vanishing markings
   27 arcs
After elimination of redundant arcs:
   # of remaining arcs: 27
ERROR/WARNING: transient initial marking. 5 markings reach it.
After the elimination of vanishing markings:
   # of remaining arcs: 27
Solving the Markov chain...
Uniformization: LTP = 104 and RTP = 230
Cumulative Uniformization: RTP = 251
...Markov chain solved
Reading the reachability graph info ...

**Reliability at time 12000 = 0.977506**

End of execution.

/\ HW3, Problem 2.c */

#include <stdio.h>
#include <math.h>
#include "user.h"

double cpufailrate=1.0/7200.0, memfailrate=1.0/14400.0, cpureprate=1.0/240.0, memreprate=1.0/120.0;

parameters()
{
  iopt(IOP_METHOD,VAL_TSUNIF);
}

enabling_type cpuRepairEnbl()
{
  if (mark("cpu_up") >= 1 && mark("mem_down") == 0) return 1;
  else return 0;
}
enabling_type memRepairEnbl()
{
    if (mark("mem_up") >= 1)
        return 1;
    else return 0;
}

rate_type cpu_repair()
{
    return(cpureprate);
}

rate_type mem_repair()
{
    return(memreprete);
}

rate_type cpu_failure()
{
    return(mark("cpu_up")*cpufailrate);
}

rate_type mem_failure()
{
    return(mark("mem_up")*memfailrate);
}

net()
{
    place("cpu_up"); init("cpu_up",3);
    place("cpu_down");
    place("mem_up"); init("mem_up",2);
    place("mem_down");

    trans("cpu_failure");
    trans("cpu_repair");
    trans("mem_failure");
    trans("mem_repair");

    iarc("cpu_failure","cpu_up");
    iarc("cpu_repair","cpu_down");

    iarc("mem_failure","mem_up");
    iarc("mem_repair","mem_down");

    oarc("cpu_failure","cpu_down");
    oarc("cpu_repair","cpu_up");

    oarc("mem_failure","mem_down");
    oarc("mem_repair","mem_up");

    enabling("cpu_repair", cpuRepairEnbl);
    enabling("mem_repair", memRepairEnbl);
}

assert()
ac_init() {pr_net_info();}
ac_reach() {pr_rg_info();}

reward_type reliability()
{
    if (mark("cpu_up") >= 1 && mark("mem_up")>=1) return(1.0);
    else return(0.0);
}
ac_final()
    time_value(12000.0);
    printf("Reliability at time 12000 = %f\n", expected(reliability));
}

Output:
SPNP Version 4.0
The analysis is starting.
The reachability graph contains:
    12 tangible markings (1 absorbing)
    0 vanishing markings
    23 arcs
After elimination of redundant arcs:
    # of remaining arcs: 23
ERROR/WARNING: transient initial marking. 5 markings reach it.
After the elimination of vanishing markings:
    # of remaining arcs: 23
Solving the Markov chain...
Uniformization: LTP = 64 and RTP = 168
Cumulative Uniformization: RTP = 186
...Markov chain solved
Reading the reachability graph info ...
Reliability at time 12000 = 0.977080
End of execution.
Problem 3

Code:

```c
#include "user.h"

double lambda_l;
double lambda_h;
double mu_l;
double mu_h;
int n;
int a;

parameters() {
    lambda_h = 5.0;
    mu_h = 5.0;
    lambda_l = 15.0;
    mu_l = 15.0;
    a = 2;
    n = a * 3;
}

enabling_type e_empty(){
    if(mark("RS") == 0){
        return(1);
    }
```
```c
} else{
    return(0);
}

/* Marking dependent firing rate */
rate_type rate_t3() { return((mark("SH")/a) * mu_h); }
rate_type rate_t4() { return((mark("SL")/a) * mu_l); }
rate_type rate_t5() { return( mark("SLL")/(a-1) * mu_l); }

net() {
/* places and intial markings: */
    place("RS");
    place("SH");
    place("SL");
    place("SLL");
    init("RS",n);

/* timed transitions and associated rates: */
    trans("T1");  rateval("T1", lambda_h);
    trans("T2");  rateval("T2", lambda_l);
    trans("T3");  ratefun("T3",rate_t3);
    trans("T4");  ratefun("T4",rate_t4);
    trans("T5");  ratefun("T5",rate_t5);
    trans("T6");  rateval("T6", lambda_h);    enabling("T6", e_empty);
    trans("T8");  rateval("T8", lambda_l);    enabling("T8", e_empty);

/* immediate transitions and associated priorities: */
    trans("T7");       priority("T7", 1);   probval("T7", 1.0);

/* input arcs */
    miarc("T1", "RS", a);
    miarc("T2", "RS",a);
    miarc("T3", "SH",a);
    miarc("T4", "SL",a);
    miarc("T5", "SLL", a-1);
    miarc("T6", "SL", a*a);
    miarc("T7", "RS",a-1);
    miarc("T7", "SLL", a-1);
    miarc("T8", "SL", (a-1)*a);

/* output arcs */
    moarc("T1", "SH",a);
    moarc("T2", "SL",a);
    moarc("T3", "RS",a);
    moarc("T4", "RS",a);
    moarc("T5", "RS", a-1);
    moarc("T6", "SLL", a);
    moarc("T6", "SH",a);
    moarc("T8", "SLL", (a-1)*a);
    moarc("T7", "SL",a);
}

assert() {/*empty*/}

ac_init() {
    pr_net_info();
}

ac_reach() {
    fprintf(stderr,"Reachability graph generated
");
    pr_rg_info();
}
```
reward_type n_HPC(){return((mark("SH")/a));}
reward_type X_LPLQC(){return(rate("T5"));}
ac_final() {
  printf("Average number of high-priority clients in the system = %f\n",
         expected(n_HPC));
  printf("The throughput of low-priority, low QoS clients = %f\n", expected(X_LPLQC));
}

Output:

SPNP Version 4.0
The analysis is starting.
The reachability graph contains:
  16 tangible markings
  9 vanishing markings
  54 arcs
Reachability graph generated
After elimination of redundant arcs:
# of remaining arcs: 54
After the elimination of vanishing markings:
# of remaining arcs: 42
Solving the Markov chain...
...Markov chain solved
Reading the reachability graph info ...
Average number of high-priority clients in the system = 0.834179
The throughput of low-priority, low QoS clients = 2.466003
End of execution.
Problem 4

Code:

```c
#include <stdio.h>
#include <math.h>
#include "user.h"

double mucl=0.1;
double mucpu=16.7;
double mudis=18.5;
int m=15;

parameters()
{
}

rate_type rate_t0()
{
    return(mark("p0")*mucl);
}
```
rate_type rate_t1()
{
    double Np=7.0;
    double B=10000000.0;
    double S=0.0000512;
    double Lp=1518;
    double A;
    double C;
    double k;
    int f;

    f=mark("p1");
    k=(double)f;
    A=pow(1.0-1.0/k,k-1);
    C=(1.0-A)/A;
    if(mark("p1")==1)
        return (1.0/((1.0/Np)*(Lp/B)));
    else
        return (1.0/((1.0/Np)*(Lp/B)+S*C));
}

probability_type proba(){return(0.5);}
probability_type probb(){return(0.5);}
probability_type probc(){return(0.5);}
probability_type probd(){return(0.5);}

net()
{
    place("p0"); init("p0",m);
    place("p1");
    place("p2");
    place("p3");
    place("pa");
    place("pb");

    trans("t0");
    trans("t1");
    trans("t2");
    trans("t3");
    trans("ta"); probfun("ta",proba); priority("ta",1);
    trans("tb"); probfun("tb",probb); priority("tb",1);
    trans("tc"); probfun("tc",probc); priority("tc",1);
    trans("td"); probfun("td",probd); priority("td",1);

    ratefun("t0", rate_t0);
    ratefun("t1", rate_t1);
    rateval("t2", mucpu);
    rateval("t3", mudis);
iarc("t0","p0");
iarc("t1","p1");
iarc("t2","p2");
iarc("t3","p3");
iarc("ta","pa");
iarc("tb","pa");
iarc("tc","pb");
iarc("td","pb");
oarc("t0","p1");
oarc("t1","pa");
oarc("t2","pb");
oarc("t3","p2");
oarc("ta","p0");
oarc("tb","p2");
oarc("tc","p3");
oarc("td","p1");
}

assert();
ac_init() {pr_net_info();}
ac_reach() {pr_rg_info();}

reward_type tput() {return(rate("t2")*0.5);}
reward_type qlength() {return(mark("p2")+mark("p3");}

ac_final()
{
    printf("The server subsystem throughput=%f\n",expected(tput));
    printf("The population at the server subsystem=%f\n", expected(qlength));
}

Output:

SPNP Version 4.0
The analysis is starting.
The reachability graph contains:
   816 tangible markings
   1360 vanishing markings
   5440 arcs
After elimination of redundant arcs:
   # of remaining arcs: 5440
After the elimination of vanishing markings:
   # of remaining arcs: 4080
Solving the Markov chain...
...Markov chain solved
Reading the reachability graph info ...

**The server subsystem throughput=1.470396**
**The population at the server subsystem=0.295990**
End of execution.