**Second set of sample exam questions**

**Gabriel Moise**

1.

(a)

void rand\_task (unsigned arg) {

message m;

int stack[MAX];

int stack\_length = 0;

// enable interrupts from hardware and connect them

while(1) {

if (stack\_length == 0) receive(HARDWARE, &m);

else receive(ANY,&m);

switch(m.m\_type) {

case INTERRUPT : {

if (! RNG\_VALRDY) break;

// clear the pending state + re-enable

if (stack\_length < MAX)

{

stack[stack\_length] = RNG\_VALUE;

stack\_length++;

}

clear\_pending(RNG\_IRQ);

enable\_irq(RNG\_IRQ);

RNG\_VALRDY = 0;

}

case REQUEST : {

message new\_m;

new\_m.m\_type = REPLY;

new\_m.m\_i1 = stack[stack\_length-1];

stack\_length --;

send(m.m\_sender,&new\_m);

}

}

}

}

unsigned randbyte(void) {

message m;

m.m\_type = REQUEST;

senderec(RANDOM,&m);

return m.m\_i1;

}

void user\_task (unsigned arg) {

unsigned result = randbyte()

serial\_printf("Random number: %d",m.m\_i1);

}

void init(void) {

serial\_init()

start(RANDOM,"Random",rand\_task,0,STACK)

start(USER,"User",user\_task,0,STACK)

}

(b) Calling randbyte several times might be too much as the speed of producing random bytes by the hardware might be too slow for the high demand and the program might need to wait too much, and in the meantime a new command might come and it might be lost.

Therefore, I changed the new\_m message to also fill in the i2,i3,i4 fields.

case REQUEST : {

message new\_m;

new\_m.m\_type = REPLY;

new\_m.m\_i1 = stack[stack\_length-1];

stack\_length --;

new\_m.m\_i2 = stack[stack\_length-1];

stack\_length --;

new\_m.m\_i3 = stack[stack\_length-1];

stack\_length --;

new\_m.m\_i4 = stack[stack\_length-1];

stack\_length --;

send(m.m\_sender,&new\_m);

}