1a

i).

TimesTwo = (in[i:0..3] -> out[i\*2] -> TimesTwo).

ii).

property MUTEX = (enter -> exit -> MUTEX).

iii).

ENTER\_PIN = ENTER\_PIN[0],

ENTER\_PIN[f:0..3] = if (f==3)

then (confiscate\_card -> STOP)

else (pin -> INP[f+1]),

INP[i:1..3] = ( ok -> ENTER\_PIN

| nok -> ENTER\_PIN[i]).

------------------------------------------------------

ENTER\_PIN = (pin -> PIN[0]),

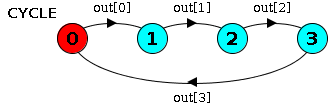
PIN[i:0..2] = (when(i<=1) nok -> pin -> PIN[i+1]

| when(i==2) nok -> confiscate\_card -> STOP

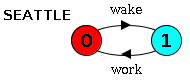
| ok -> ENTER\_PIN

1b

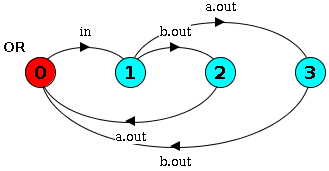
i).



ii).



iii).



1c

i).

Check [Exam 2016](https://docs.google.com/document/d/1I1efEW9CBGvTTqb09vPeT3811INLV-Xu19olqOfCpQg/edit?usp=sharing) - 2a, i)

ii).

Check [Piazza post](https://piazza.com/class/itv4vb3e5wm8i?cid=43).

2a

VM(N=0) = OPEN\_VM[N],

OPEN\_VM[left:0..N]

= (when (left > 0) get -> OPEN\_VM[left-1]

|when (left < N) put -> OPEN\_VM[left+1]

|close -> CLOSED\_VM[left]),

CLOSED\_VM[left:0..N]

= (when (left < N) put -> CLOSED\_VM[left+1]

|when (left==N)service -> OPEN\_VM[left]).

2b Also discussed in a [Piazza post](https://piazza.com/class/itv4vb3e5wm8i?cid=31).

||GOLFCLUB = (Members:MEMBER || dave:TECHNICIAN ||

{Members, dave}::

(club:VM(MAXclubs) || ball:VM(MAXballs))

).

2c Similar solution with Jeff’s comments available on [Piazza](https://piazza.com/class/itv4vb3e5wm8i?cid=37).

public class VM {

private int left;

private bool open;

private int max;

public VM(int max) {

left = max;

this.max = max;

open = true;

}

public synchronized void get() throws InterruptedException {

while (!(left>0) || !open) { wait() };

Left--;

notifyAll();

}

public synchronized void put() throws InterruptedException {

while (left==max) { wait() };

left++;

notifyAll();

}

public synchronized void close() {

while (!open) wait();

open = false;

}

public synchronized void service() {

while (open || left != max) wait();

open = true;

notifyAll();

}

}

public class Member implements Runnable {

private VM club;

private VM ball;

public Member(VM club, VM ball) {

this.club = club;

this.ball = ball;

}

@Override

public void run() {

while(true) {

try {

club.get(); ball.get();

sleep(500); // play

club.put(); ball.put();

} catch (InterruptedException e){}

}

}

void build() {

VM club = new VM(MAXclubs);

VM ball = new VM(MAXballs);

Member alice = new Member(club, ball);

Member bob = new Member(club, ball);

Member cas = new Member(club, ball);

Technician dave = new Technician(club, ball);

}

2d

If dave calls close() twice on the same VM without servicing the machine, the system will deadlock. However, I am not sure whether this is the answer they are looking for

alice.club.get

bob.club.get

Dave.club.close

Dave.ball.close

dave.club.service (dave *blocked waiting on all clubs to be returned)*

alice.ball.get (alice *blocked waiting on dave to service ball)*

Bob.ball.get (bob *blocked waiting on dave to service ball)*

Cas.club.get (cas *blocked waiting on someone to return a club)*

DEADLOCK (*all threads blocked as described above)*

TECHNICIAN = (club.close -> ball.close -> ball.service -> club.service -> TECHNICIAN) + All.

Prevents circular wait deadlock condition.

Separate maintenance of club VM and Ball VM so dave can only service one machine at a time.

2e

progress Practice = {alice.play}

No, in the original model, this property is not violated.