Problem 1

Part a

```
chan in(int)
chan out1(int)
chan out2(int)

process Partition:
    int v;
    receive in(v)
    while( !empty(in) ):
        int next;
        receive in(next)

        if next <= v:
            send out1(next)
        else:
            send out2(next)

    send out1(v)
    send out2(EOS)
```

We can make the following conclusions about the values present in the values entering and exiting the process:

out1: $i \leq v, \forall i \in \textbf{out1}$ **out2**: $i > v, \forall i \in \textbf{out2}$

in: $v \in \textbf{in} \ \wedge \textbf{in} = \textbf{out1} \cup \textbf{out2}$

Part b

This Partition functions identically to the partition algorithm that is used in quicksort. Using this, we could repeatedly partition until we are only given two elements. In this case, we would need at most $log(n)$ servers to handle all of the partitions.

Problem 2

```
channel toServer(command, philosopher, left, right)
bool chopsticks[n]

process Server {
    while( true ) {
        receive philosopher(command, philsopher, l, r)

        if( command == PICK-UP ) {
            if(chopsticks[l] and chopsticks[r]) {
                chopsticks[l] = chopsticks[r] = 0
                send philosopher()
            } else {
                # Put the client on a queue
            }
        } else if( command == PUT-DOWN ) {
            chopsticks[l] = chopsticks[r] = true
            # Get philsopher waiting on l
            send philosopherWaitingOnL()

            # Get philsopher waiting on r
            send philosopherWaitingOnR()
        }
    }
}
```

A philsopher would interact with the server like so:

```
# Pick up the chopsticks
send toServer(PICK-UP, myChan, leftChop, right Chop)
receive myChan()

# User chopsticks

# Put down chopsticks
send toServer(PUT-DOWN, myChan, leftChop, right Chop)
```

Problem 3

```
chan fromA(int, int, channel)
int numMet[n]

process Server {
    while( (not empty(as)) and (not empty(bs)) ) {
        int who, myId;
        chan response;

        receive toServer(who, myId, response)

        if( who == A ) {
            if( numMet[myId] == 2 ) {
                send response(LEAVE)
            } else {
                numMet[myId]++
                send response(CONTINUE)
            }
        } elif( who == B ) {
            send response(LEAVE)
        }
    }
}
```