ASSIGNMENT #2

**Question 1**

PHIL = (think -> PHIL | get\_cookie -> eat\_cookie -> PHIL | get\_cola -> drink\_cola -> PHIL)

SERVANT = (refill\_cookies -> SERVANT | refill\_cola -> SERVANT)

COOKIE\_DISP(K = M) = COUNT\_COOKIES[K]

COUNT\_COOKIES[i: 0..M] =

(when (i == 0) refill\_cookies -> COUNT\_COOKIES[M]

|give\_cookie -> COUNT\_COOKIES[i-1])

COLA\_DISP(K = N) =

(when (i == 0) refill\_cola -> COUNT\_COLA[N]

|give\_cola -> COUNT\_COLA[i-1])

||DINERS(N = k) = (forall [i: 1..N]

(phi[i] : PHIL || SERVANT || phil[i] :: COOKIE\_DISP(M) || phil[i] :: COLA\_DISP(N))/{get\_cookie/give\_cookie}

**Question 2**

DOOR =

DIRECTOR = (open -> DIRECTOR

|closed -> DIRECTOR)

**Question 3**

const N = 4

range T = 0..N

CARPARKCONTROL(N=4) = SPACES[N]

SPACES[i:0..N] = (when (i>0) arrive->SPACES[i-1]

|when(i<N) depart->SPACES[i+1]

)

ARRIVALS = (arrive -> ARRIVALS)

DEPARTURES = (depart -> DEPARTURES)

||CARPARK = (ARRIVALS || CARPARKCONTROL(4) || DEPARTURES)

property OVERFLOWN = (arrive -> CARS[1])

CARS[i: 0..N] = (when (i < N) arrive -> CARS[i+1]

|when (i == 1) depart -> OVERFLOWN

|when (i > 1) depart -> CARS[i-1])

progress CAR\_ARRIVAL = {arrive}

**Question 4**

**Question 5**

FORK = (get -> put -> FORK)

PHIL = (when (i = 1 ∨ i = 3 ∨ i = 5) think -> left.get ->

right.get -> eat -> left.put -> right.put -> PHIL

|when (i = 2 ∨ i = 4) think -> right.get ->

left.get -> eat -> right.put -> left.put -> PHIL)

|| DINERS(N = 5) = (forall[i : 1..N]

(phil[i] : PHIL k {phil[i].right, phil[i ⊕ 1].left} :: FORK))