1 micro

50 personas de capacidad

1 vendedor

-C clientes (C>50) compran pasajes por orden de llegada

-Si hay lugares, le indica el número de asiento.

-Si no hay lugar se retira.

Sem fila= 0;

Sem pasajeros = 0;

ColaC cola;

Sem mCola= 1;

Sem Siguiente[C] =0;

Atención[C];

**Process Cliente [id=0..C-1]**

{ P(mCola)

Push (cola, id)

V(mCola)

V(fila) //

P(siguiente[id])

If (atención[id]!=-1){

V(subio);}

Else { //se retira sin pasaje}

}

**Process Vendedor** {

Vendidos=0;

Int i, C;

for(i=0;i<C;i++){

P(fila)

P(mCola)

Pop(cola,idC)

V(mCola)

If (vendidos<50){

Vendidos++;

Atención[idC]= asignarAsiento();

Else{

Atencion[idC]=-1}

P(siguiente[idC])}}}

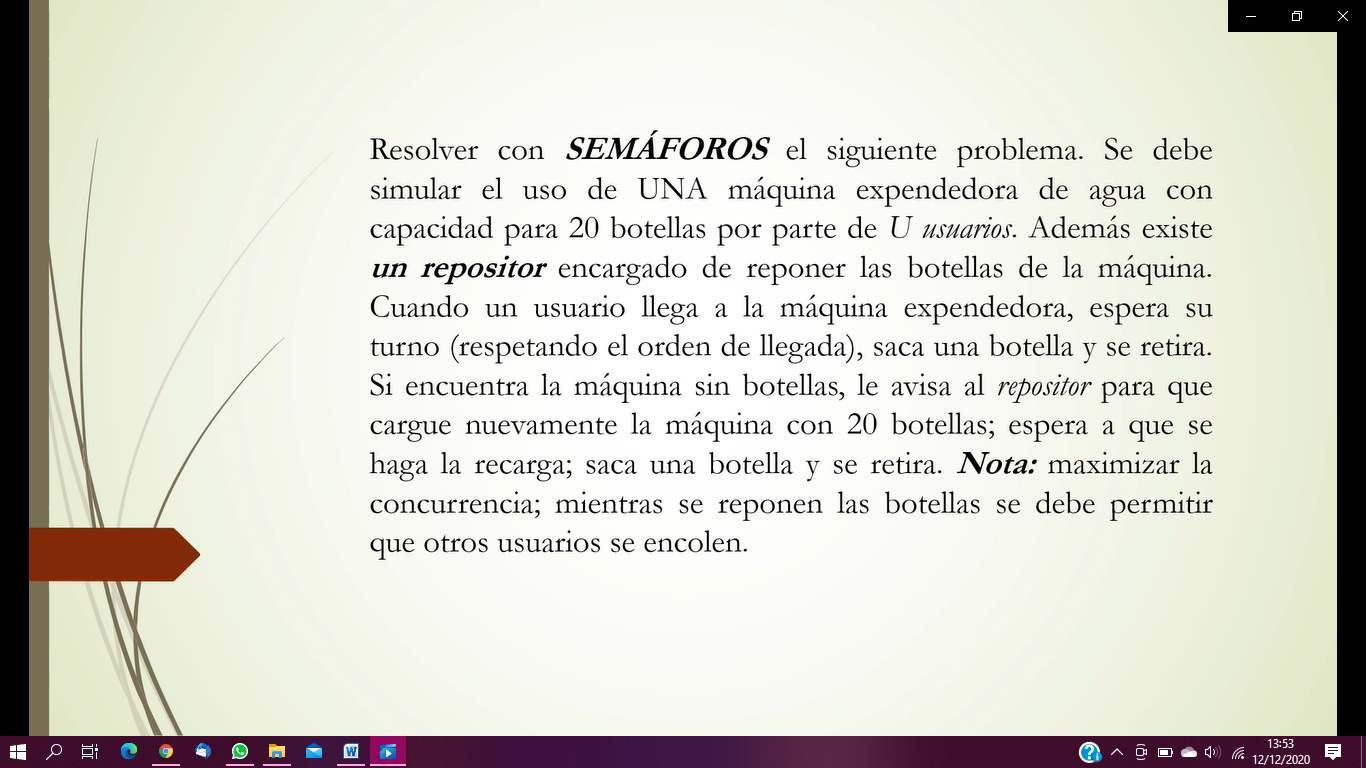
**Process Micro** {

Int i;

For (i=0;i<50;i++){

P(subio);}

//comienza el viaje }



1 maq expendedora de agua

Capacidad 20 botellas

U usuarios

1 repositor que repone las botellas

Queue colaUsuarios;

Sem mutex=1;

Bool libre;

Cant=20;

Int Ultimo=-1;

Sem turno[U]=([U]0);

Botellas[U]= ([U]0);

Sem habilitado=0;

Sem reponer=0;

**Process Usuario[id=0..U-1]**{

P(mutex)

If(not libre){

Push(colaUsuarios, id)

V(mutex)

P(turno[id]);}

Else{

libre=false;//la ocupo

V(mutex);}

If (cant==0){

V(reponer);

P(habilitado)}

//sacar botella

Cant--;

P(mutex)

If(empty(colaUsuarios)){libre= true;}

Else {pop(colaMaq, aux):

V(espera[aux])}

V(mutex)

**Process Repositor**{

While (true){

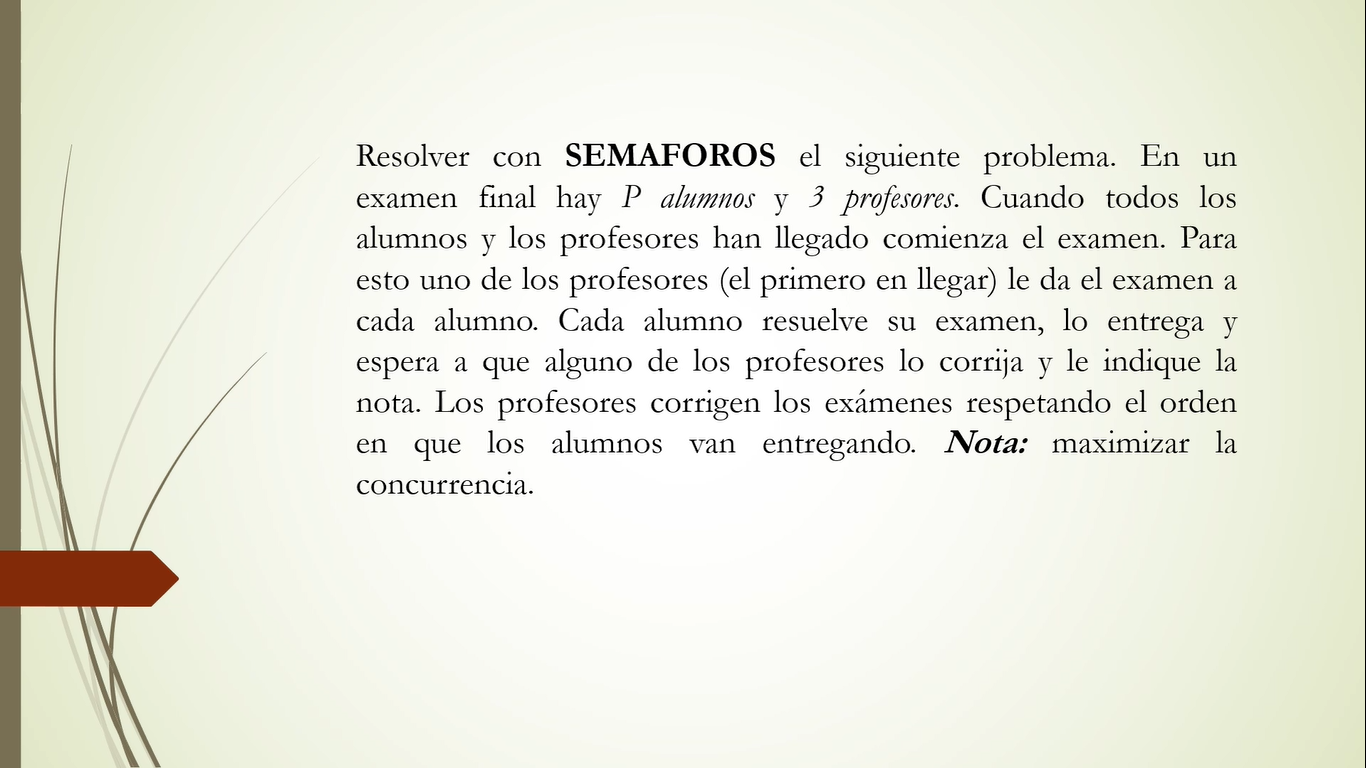
P(reponer)

For (i=1;i<20;i++){ cargarBotella();}

Cant=20;

V(habilitado)

}}



Sem llegue=1;

Sem mutex=1;

Sem primerProfesor=0;

Sem reparto[P]=[P]0;

Sem esperaNota[P]=[P]0;

Sem realizado:=0;

Int notas[P];

Int Cant=0;

Int primerP=-1;

Queue colaExamenes;

Text Exámenes[P]

**Process Alumno [id=1..P-1]**{

Real nota;

Text E;

P(llegue)

Cant++;

If(cant=P+3){V(primerP)}

V(llegue)

P(reparto[id])

E=realizar(exámenes[id]);

P(mutex)

Push(colaExamenes,(id,E));

V(mutex)

V(realizado);

P(esperaNota[id])

Nota= notas[id]}

**Proccess Profesor [id=0..2]**{

Int idC;

Text E;

P(llegue)

Cant++;

If(cont=P+3){V(primerProfesor)}//si llegaron todos

If(primerP=-1){primerP=id}//si soy el primer profe en llegar

V(llegue)

If(primerP==id){//si fui el primero, reparto los exámenes

P(primerProfesor)

For(i=1;i<P:i++){

Exámenes[i]= generarExamen()

V(reparto[i])}}

While (true){

P(realizado)

P(mutex)

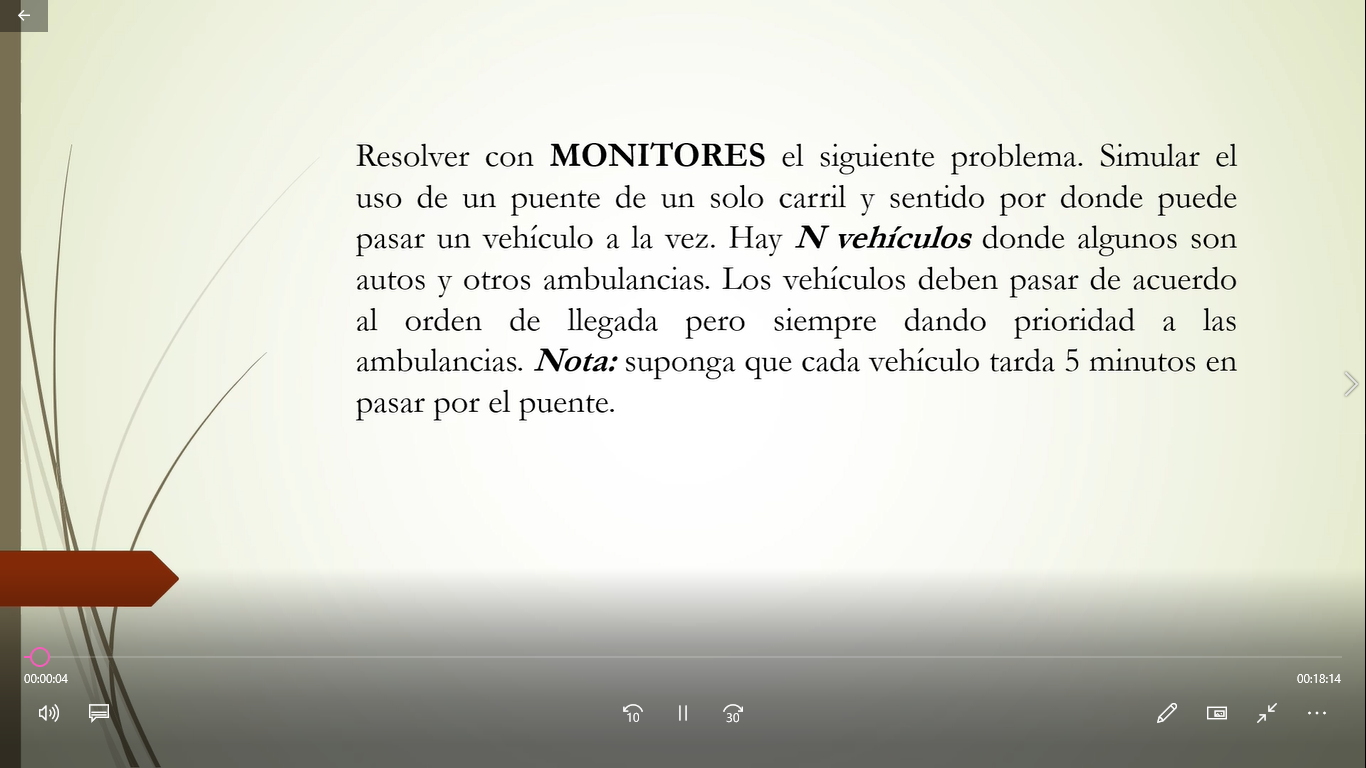
Pop(colaExamenes,(idC,E));

V(mutex)

Notas[idC]=coregirExamen(E)

V(esperarNota[idC])

}}



Process vehiculo [id=0..N-1]{

Bool esAmbulancia= …;

If (esAmbulancia){Asministrador.PasoAmbulancia()}

Else{Administrados.PasoAuto()}

//cruza el puente

Delay(300);

Administrador.Salir();

}

Monitor administrador{

Bool libre= true;

Int canrAmb=0;

Int cantAutos=0;

Cond esperaAuto,esperaAmb;

Process PasoAmbulancia(){

If (libre){libre= false}}

Else{

cantAmbulancia++;

wait(esperaAmbulancia)}

Process PasoAuto(){

If (libre){libre= false}}

Else{

cantAuto++;

wait(esperaAuto)}

Process Salir(){

If (cantAmbulancia>0){

cantAmbulancia - -

signal (esperaAmbulancia)}

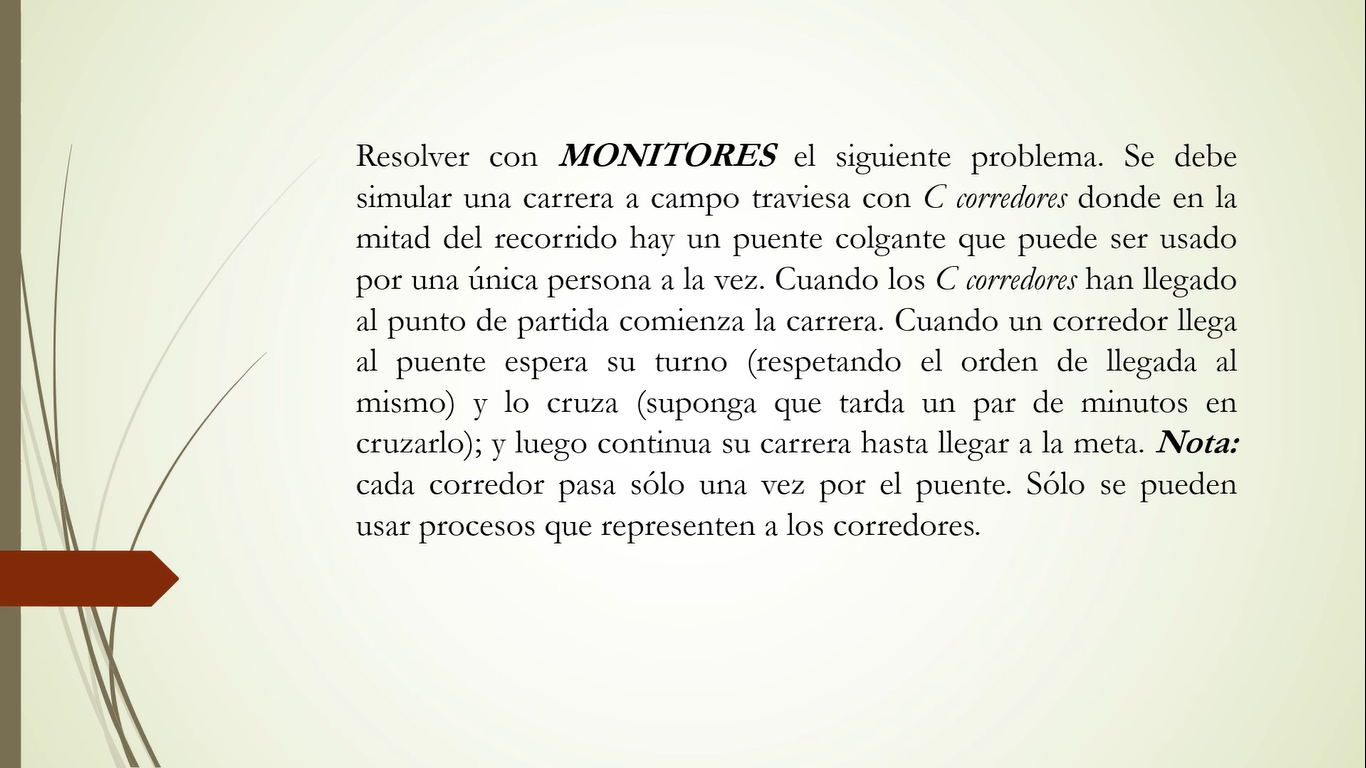
else{ if(cantAutos>0){

cantAutos- -;

signal(esperaAuto)}

else{libre=true;}}

}



Process Corredor[id=1..C]{

Largada.EsperarATodos();

//comienza la carrera

AdminP.SolicitarPaso();

//cruzar puente

Delay(60);

Admin.Liberar();

//continua hasta la meta

}

Monitor Largada{

Cond autos; Juan

Cant=0;

Process EsperarATodos{

Cant++;

If (cant<C){wait (autos)}

Else {

Signal\_all(autos)}}}

Monitor AdminP{

Bool libre=true;

Cond paso;

cantAuto:=0;

Process SolicitarPaso(){

If (libre){libre= false}

Else{cantAuto++;

wait(paso)}}

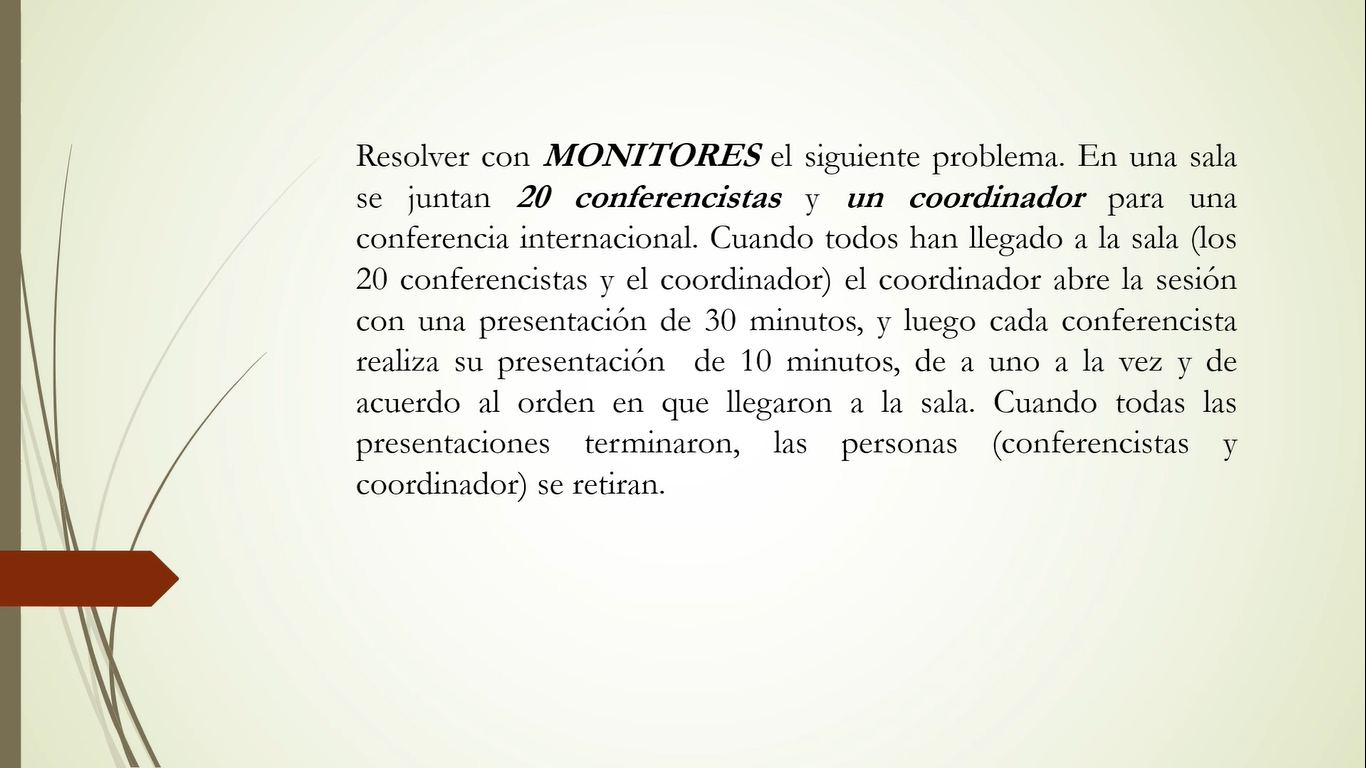
Process Liberar{

If (cantAuto>0){cantAuto - -;

signal(paso)}

Else {libre=true}}

}



Process Conferencista[id=1..20]{

Sala.Llegada();

//presentación

Delay (600);

Sala.Fin();

}

Process Coordinador{

Sala.LlegadaCoord();

//presentación

Delay(1800);

Sala.Fin()

}

Monitor Sala{

Terminaron=0;

Cant:=0;

Cond turnoConf, turnoCoord,esperaFin;

Process Llegada{

Cant++;

If( cant=20){ signal(turnoCoord) }

wait(turnoConf);

Process LlegadaCoord{

If( cant<20){wait(turnoCoord)}}

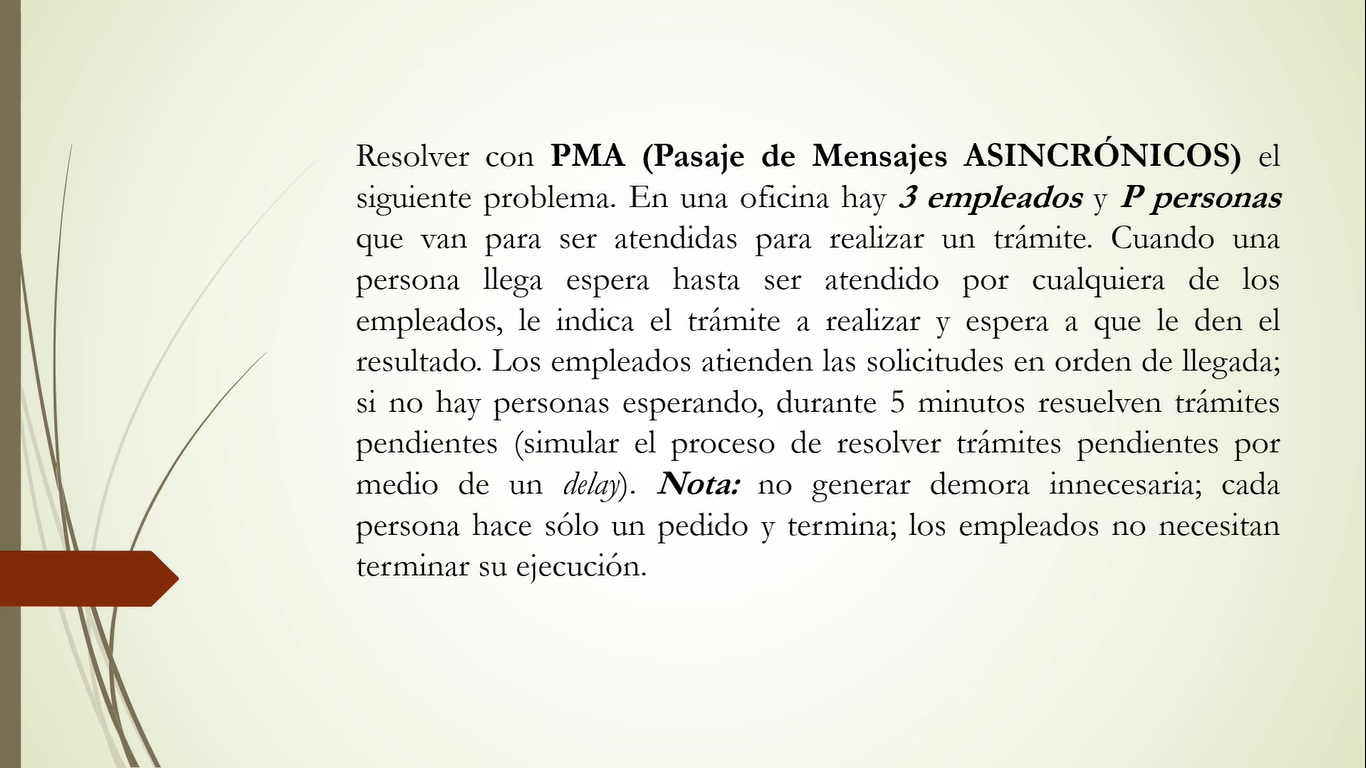
Process Fin{

Terminaron++;

If (terminaron=21){signal\_all(esperaFin)}

Else {signal(turnoConf);

Wait(esperaFin)}}}}



Chan atención(int)

Chan tramite[3]

Chan resul[P]

Chan turno[P];

Chan pedido(int);

Chan siguiente[3]()

Process Persona [id=1..P]{

T=tramiteARealizar();

Send atención(id);

Receive turno[id](idE);

Send trámite[idE](T);

Receive resul[id](res)

}

Process Empleado [id=1..3]{

While(true){

Send Pedido(id);

Receive Siguiente[id](next);

If(next!= -1){Send turno[next](id)

Receive trámite[next](T)

R=resolverTramite()

Send resul[next](R)}

Else {delay 300}

}}

Process Coordinador{

Int Next, idE;

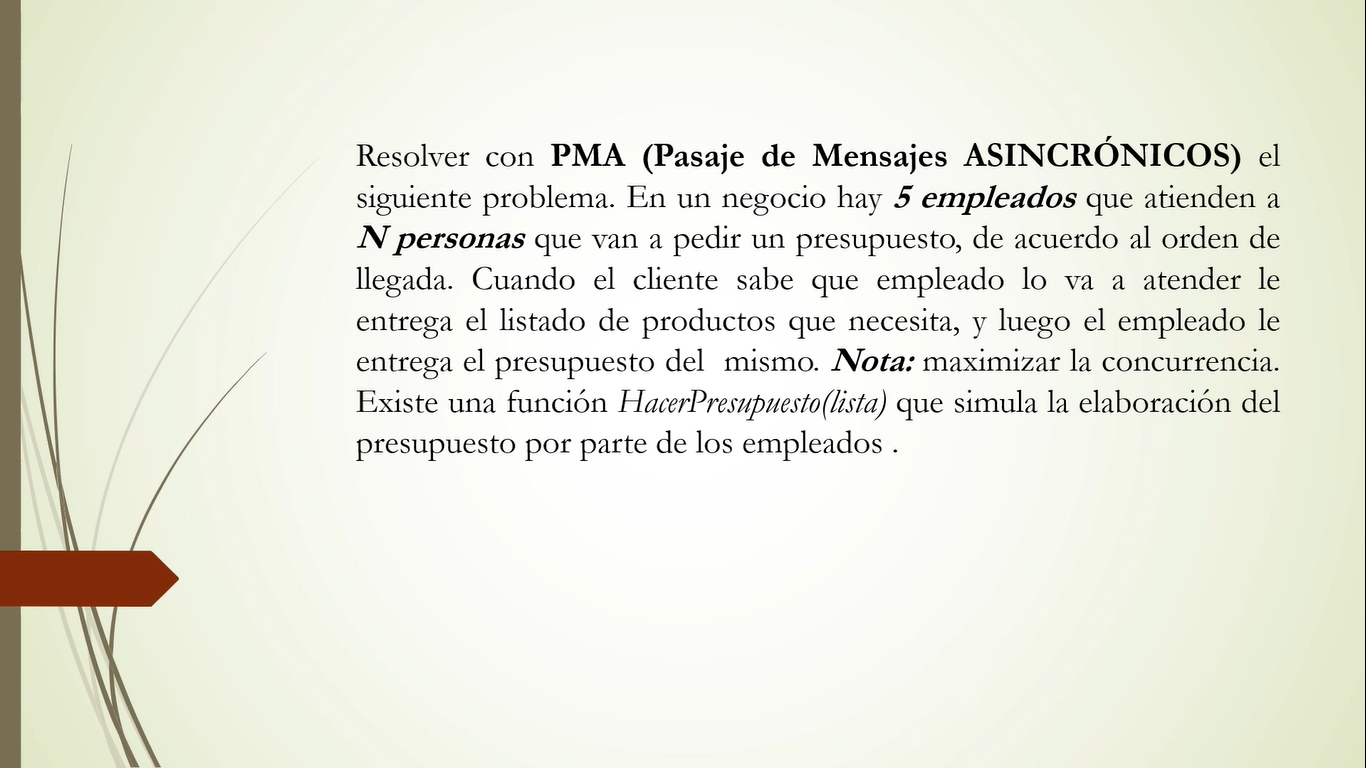
While(true){

Receive pedido(idE);

If(not Empty atención){receive atención(Next);}

Else{next= -1}

Send siguiente[idE] (Next);}}



Chan atención(int)

Chan tramite[3]

Chan resul[P]

Chan turno[P];

Chan pedido(int);

Chan siguiente[3]()

Process Persona [id=1..P]{

L=generarLista();

Send atención(id);

Receive turno[id](idE);

Send lista[idE](L);

Receive presupuesto[id](res)

}

Process Empleado [id=1..3]{

While(true){

If (not Empty (atención)){

Receive atención(idC);

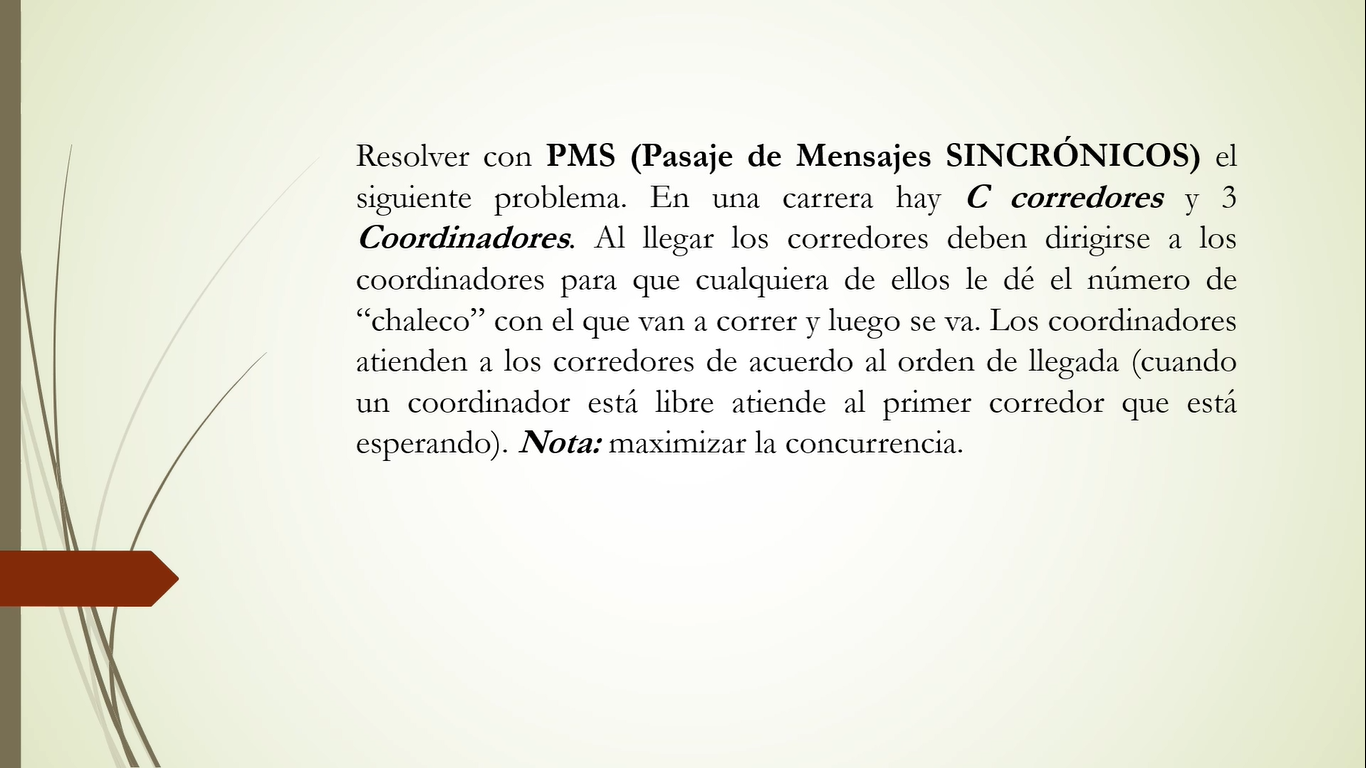
Send turno[idC](id)

Receive lista[idC](L)

P=hacerPresupuesto()

Send resul[idC](P)}

}}



Process Corredor[id=1..C]{

Admin!Llegada(id)

Coordinador[\*]?chaleco(n);}

Process Coordinador[id=1..3]{

While true{

Admin!pedido(id)

Admin?siguiente(idC);

C=numeroDeChaleco();

Corredor[idC]!chaleco (C)}}

Process Admin{

Do Corredor[\*]?Llegada(idC) -> push (fila,idC)

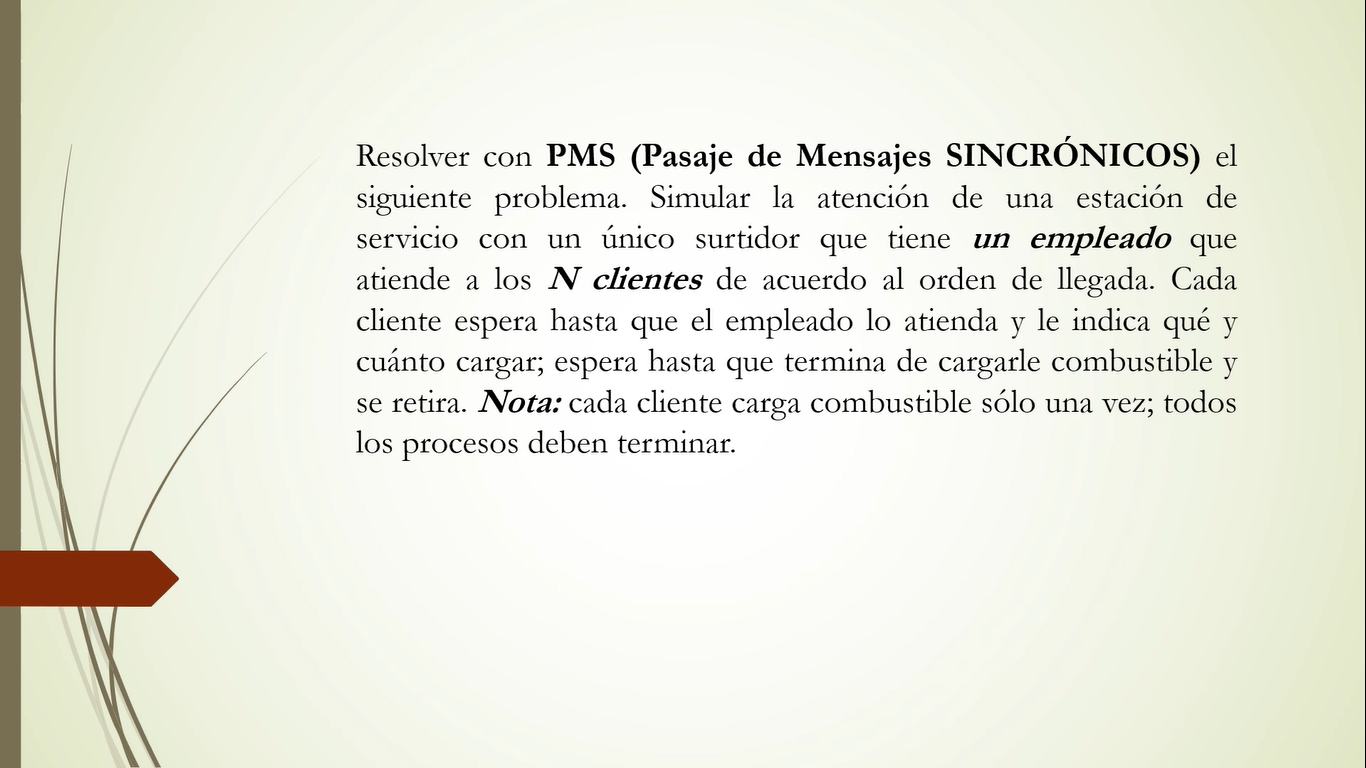
□ not empty (fila); Coordinador[\*]?pedido(idCo)->

Pop (fila, idC)

Coordinador[idCo]!siguiente(idC);

od

}



Process Cliente[id=1..N]{

Admin!Llegada(id)

Empleado?pasar();

Empleado!cargar(combustible, cantidad);

Empleado?salir();}

Process Empleado{

Cant=0;

While (cant<N){

Admin!pedido(id)

Admin?siguiente(idC, N);

Cliente[idC]!pasar();

Cliente[idc]?cargar(tipo, cant);

//se realiza la carga

Cliente[idC]!salir }}

Process Admin{

Atendidos=0;

While(atendidos<N)

Do Cliente[\*]?Llegada(idC) -> push (fila,idC)

□ not empty (fila); Empleado?pedido->

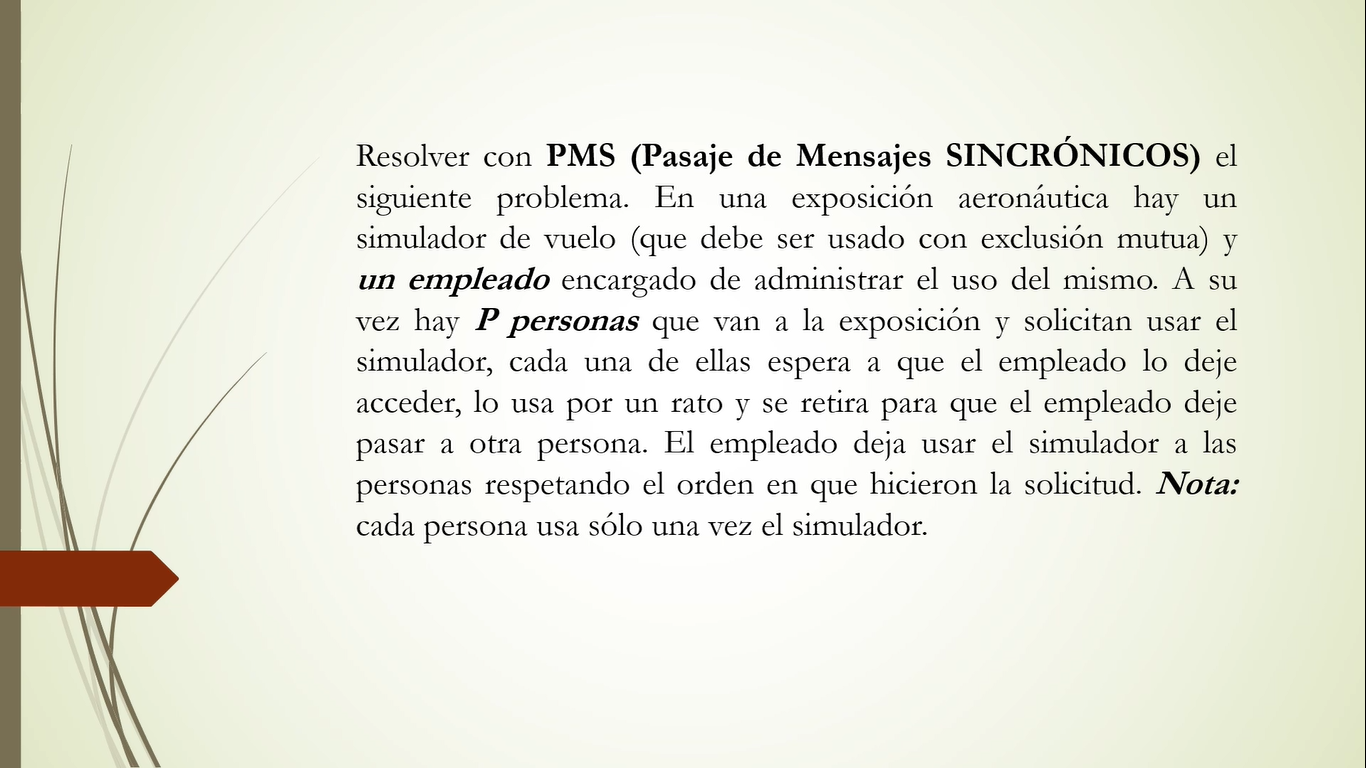
Atendidos++;

Pop (fila, idC)

Empleado!siguiente(idC, atendidos);

od

}



Process Persona[id=1..P]{

Empleado!Hola(id);

Empleado?Turno();

//usa el simulador

Empleado!MeVoy();

}

Process Empleado{

Cola fila;

Int idP;

Libre=true;

Do Persona[\*]?Hola(idP)->if (not libre){ push (fila,idP)}

Else{libre= false;

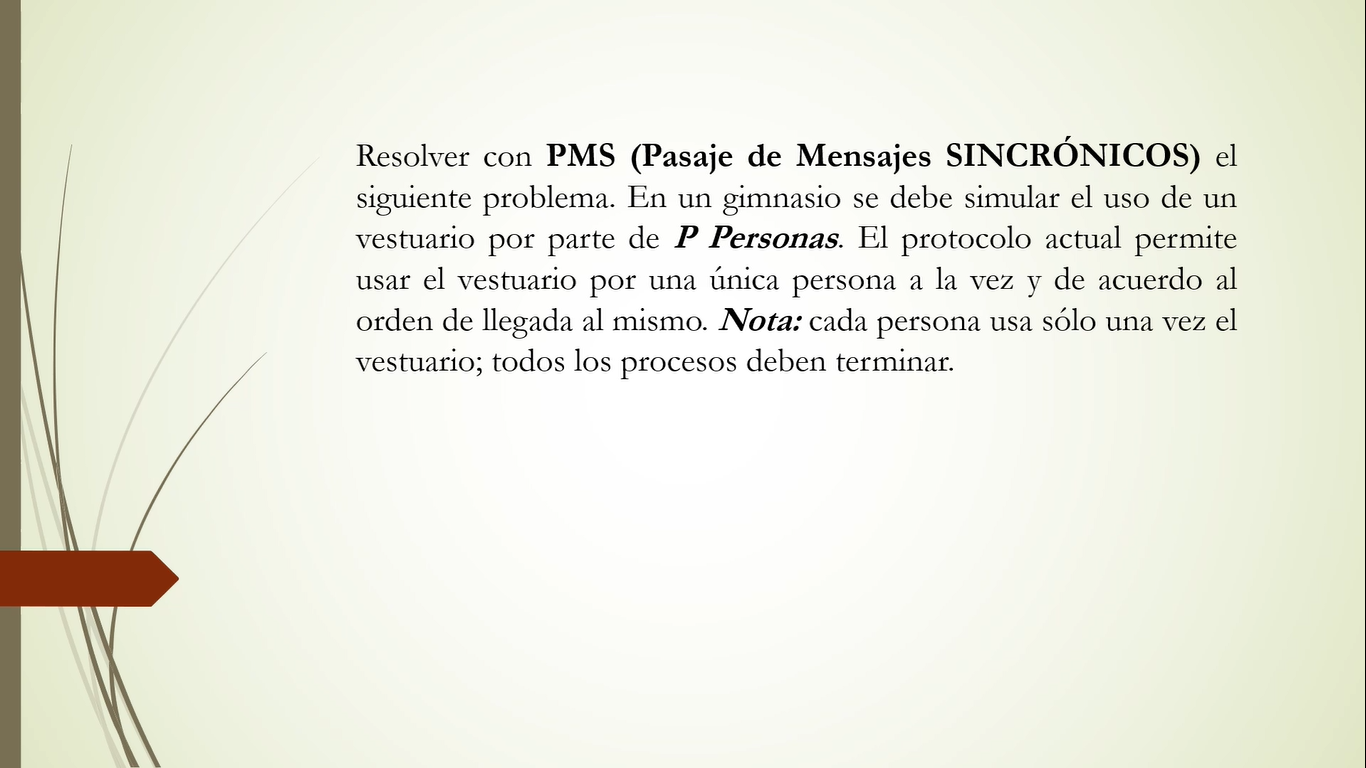
Persona[idP]!Turno()}

□Persona[\*]?MeVoy()->if (not empty fila){pop (fila,idP)

Persona[idP]!Turno()}

Else{libre=true}

od}



Process Persona [id=1..P]{

Admin!Hola(id);

Admin?Pasar();

//usar el vestuario

Admin!YaSali();

}

Process Admin{

Bool Libre=true;

Cola fila;

Int Cant=0;

Int idP;

While (cant<P){

If Persona[\*]?Hola(idP)->if (not libre){push (fila, idP)}

Else{libre=false;

Persona[idP]!pasar()}

□Persona[\*]?YaSali()->cant++

if (not empty(fila){pop (fila, idC);

Persona[idP]!pasar()}

Else{libre=true}}}