

1-

Process Persona[id=0..P]{

While true{

Empleado!DejarPaquete(destino,paquete)}}

Process Camion[id=0..1]{

While true{

Int cant;

Empleado!pedido(id)

Empleado?cantPaquetes(cant)

For(int i=1;i<cant; i++){

Empleado?pedido(paquete);

//cargar paquete}

//emprender el viaje}}

Process Empleado{

Text d;

Paquete p;

Cola paqAlNorte;

Cola paqAlSur;

Int norte,sur=0;

Do Persona[\*]?DejarPaquete(d,p)->if(d=norte){push(paqAlNorte, paquete;

Norte++;)}

Else{ push(paqAlSur, paquete

Sur++;)}

□not empty (paqAlNorte); Camion[0]?pedido()->

Camion[0]!cantPaquetes(Norte)

While(not empty(paqAlNorte)){

Pop(paqAlNorte,paquete);

Camion[0]!pedido(paquete)

}

□ not empty (paqAlSur);Camion[1]?pedido->

Camion[1]!cantPaquetes(Sur)

While(not empty(paqAlSur)){

Pop(paqAlSur,paquete);

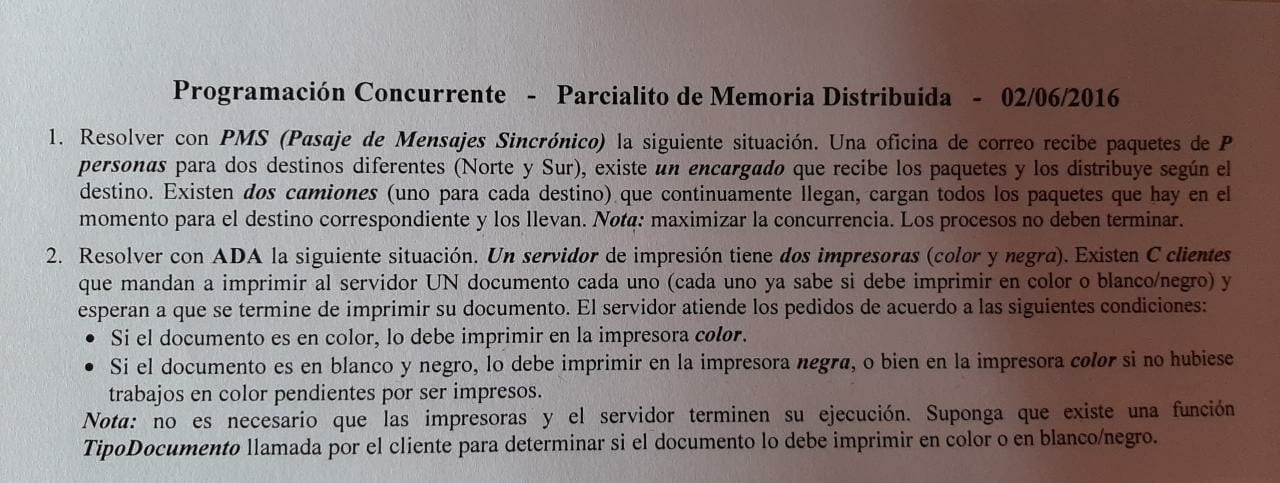
Camion[1]!pedido(paquete)}

}

**Otra solución al problema**

| Process Persona [1..p]{  while (true) {  Encargado ! enviarPedido (destino, pedido);  } }  Process Camion [2]{ *//El enunciado no dice nada sobre la capacidad de los camiones, solamente dice que cuando llega, carga TODOS los pedidos a ese destino y se va*  var  string destino; *//Camion con id 1 es al norte, camión con id 2 es al sur*  string [n] paquetes; *//Paquetes a entregar a ese destino*   while (true) {  Encargado ! ListoParaPartir (destino,paquetes);  if (!empty (paquetes))  delay *//viaja a destino, entrega los paquetes y vuelve*   }  }  Process Encargado {  var   queue pedidosParaNorte; queue pedidosParaSur;  string pedido; string destinoPedido;   while (true) {  if  [] Persona[\*] ? enviarPedido (destinoPedido, pedido) ----->  if (destinoPedido==1) {  pedidosParaNorte.push (pedido)  } else {  pedidosParaSur.push (pedido)  }  [] (!empty (pedidosParaNorte)) Camion[1] ? ListoParaPartir (destino, paquetes) ----->  while (!empty (pedidosParaNorte)) {  pedidosParaNorte.push(pedido);  paquetes.add(pedido)  }   [] (!empty (pedidosParaSur)) Camion[2] ? ListoParaPartir (destino, paquetes) ----->   while (!empty (pedidosParaSur)) {  pedidosParaSur.push(pedido);  paquetes.add(pedido)  }  fi  }  } |
| --- |

|  |
| --- |
|  |



Procedure impresión is

TASK servidor is

ENTRY ListoColor(D:OUT texto; idC: OUT int)

ENTRY ListoNegro(D:OUT texto; idC: OUT int)

ENTRY ImprimirColor(D:IN texto; idC: IN int)

ENTRY ImprimirNegro(D:IN texto; idC: IN int)

End servidor;

TASK impresoraColor ;

TASK impresoraNegro;

TASK TYPE cliente is

ENTRY Ident (Pos: IN integer)

ENTRY Finalizada (I:IN impresion)

Clientes: ARRAY(1..C) of cliente;

TASK BODY servidor is

begin

text D;

int idC;

loop

SELECT

Accept ListoColor (D:OUT texto; idC:OUT int) do

SELECT Accept imprimirColor (doc:IN texto; id:IN int)do

D:= doc;

IdC:= id;

OR WHEN (ImprimirColor count=0) =>

Accept imprimirNegro (doc:IN texto; id:IN int)do

D:= doc;

IdC:= id;

OR

Accept ListoNegro (D:OUT texto; idC:OUT int) do

Accept imprimirNegro (doc:IN texto; id:IN int)do

D:= doc;

IdC:= id;

end loop

End servidor;

TASK BODY impresoraColor is

text D;

impresion I;

int idC;

loop

Servidor.ListoColor(D,idC)

I=imprime(D);

Cliente(idC).Finalizado(I)

End loop

End impresoraColor;

TASK BODY impresoraNegro is

text D;

impresion I;

int idC;

loop

Servidor.ListoNegro(D, idC)

I=imprime(D);

Cliente(idC).Finalizado(I)

End loop

End impresoraNegro;

TASK BODY cliente is

id:integer

Documento: texto;

Tipo:texto;

Begin

Accept Indent (pos: IN integer)do

id:= Pos

end Indent

Tipo=tipoDocumento();

If (tipo=negro) Servidor.ImprimirNegro(Documento, id)

Else Servidor.ImprimirColor(Documento, id)

Accept Finalizado(I:IN impresion )

End cliente;

BEGIN

for i in 1..N loop

clientes(i).Indent(i);

end loop;

END IMPRESION