Programación Concurrente -Plan2003 – 20/11/08 1er fecha Tema 1

1. Implementar utilizando monitores el problema de lectores/escritores de acuerdo a las siguientes pautas:

   a. El monitor de debe definir sólo tres procedimientos.
   b. El monitor debe contener una única variable condition.
   c. Los lectores no pueden quedar dormidos dentro del monitor, ni realizar busy waiting.

2. Una pizzería hay 5 maestros pizzeros los cuales elaboran pizzas de la siguiente manera. Cuando un maestro está disponible realiza la pizza y la deposita en una bandeja de pizzas listas. El pizzero trabaja 3 horas y luego se retira.

   NOTAS:
   En el caso que durante la elaboración de la pizza transcurren las 3 horas, el pizzero termina esa pizza y luego se retira.
   No se puede suponer nada sobre el tiempo que requiere hacer la pizza.
   los alumnos que rinden REDUCIDO pueden elegir para implementar el ejercicio el modelo de pasaje de mensajes (sincrónico o asincrónico) o ADA.
   los alumnos que rinden COMPLETO deben implementar el ejercicio con ADA.

3. SOLO PARA LOS ALUMNOS QUE RINDEN COMPLETO
   Un banco decide entregar promociones a sus clientes por medio de su agente de prensa, el cual lo hace de la siguiente manera: el agente debe entregar 50 premios entre los 1000 clientes, para esto, obtiene al azar un número de cliente y le entrega el premio, una vez que este lo toma continúa con la entrega.
   NOTAS:
   Cuando los 50 premios han sido entregados el agente y los clientes terminan su ejecución.
   No se puede utilizar una estructura de tipo arreglo para almacenar los premios de los clientes.
   Resolver con Semáforos.

Programación Concurrente -Plan2003 – 20/11/08 1er fecha Tema 2

1. Implementar utilizando monitores el problema de lectores/escritores de acuerdo a las siguientes pautas:

   a. El monitor de debe definir sólo tres procedimientos.
   b. El monitor debe contener una única variable condition.
   c. Los lectores no pueden quedar dormidos dentro del monitor, ni realizar busy waiting.

2. Una pizzería hay 5 maestros pizzeros los cuales elaboran pizzas de la siguiente manera. Cuando un maestro está disponible realiza la pizza y la deposita en una bandeja de pizzas listas. El pizzero trabaja 3 horas y luego se retira.

   NOTAS:
   En el caso que durante la elaboración de la pizza transcurren las 3 horas, el pizzero termina esa pizza y luego se retira.
   No se puede suponer nada sobre el tiempo que requiere hacer la pizza.
   los alumnos que rinden REDUCIDO pueden elegir para implementar el ejercicio el modelo de pasaje de mensajes (sincrónico o asincrónico) o ADA.
   los alumnos que rinden COMPLETO deben implementar el ejercicio con ADA.

3. SOLO PARA LOS ALUMNOS QUE RINDEN COMPLETO
   Un banco decide entregar promociones a sus clientes por medio de su agente de prensa, el cual lo hace de la siguiente manera: el agente debe entregar 50 premios entre los 1000 clientes, para esto, obtiene al azar un número de cliente y le entrega el premio, una vez que este lo toma continúa con la entrega.
   NOTAS:
   Cuando los 50 premios han sido entregados el agente y los clientes terminan su ejecución.
   No se puede utilizar una estructura de tipo arreglo para almacenar los premios de los clientes.
   Implementar el ejercicio con Semáforos

1. Implementar utilizando monitores el problema de lectores/escritores de acuerdo a las siguientes pautas:

    a. El monitor de debe definir sólo tres procedimientos.
    b. El monitor debe contener una única variable condition.
    c. Los lectores no pueden quedar dormidos dentro del monitor, ni realizar busy waiting.

2. Una pizzería hay 5 maestros pizzeros los cuales elaboran pizzas de la siguiente manera. Cuando un maestro está disponible realiza la pizza y la deposita en una bandeja de pizzas listas. El pizzero trabaja 3 horas y luego se retira.

    NOTAS:
    En el caso que durante la elaboración de la pizza transcurren las 3 horas, el pizzero termina esa pizza y luego se retira.
    No se puede suponer nada sobre el tiempo que requiere hacer la pizza.
    los alumnos que rinden REDUCIDO pueden elegir para implementar el ejercicio el modelo de pasaje de mensajes (sincrónico o asincrónico) o ADA.
    los alumnos que rinden COMPLETO deben implementar el ejercicio con ADA.

3. SOLO PARA LOS ALUMNOS QUE RINDEN COMPLETO
    Un banco decide entregar promociones a sus clientes por medio de su agente de prensa, el cual lo hace de la siguiente manera: el agente debe entregar 50 premios entre los 1000 clientes, para esto, obtiene al azar un número de cliente y le entrega el premio, una vez que este lo toma continúa con la entrega.
    NOTAS:
    Cuando los 50 premios han sido entregados el agente y los clientes terminan su ejecución.
    No se puede utilizar una estructura de tipo arreglo para almacenar los premios de los clientes.

1- Dado el siguiente enunciado y código:

Enunciado: Se tienen M campamentistas, cada niño debe obtener 12 elementos; para esto cada campamentista pide un elemento en forma consecutiva (primero el elemento uno, después el dos, y así sucesivamente) al encargado y espera por el mismo un máximo de 7 minutos, para lo cual utiliza una tarea reloj la cual controla que no pasen 7 minutos para el pedido de cada elemento. Al finalizar el campamentista informa qué ejemento /s pudo conseguir y cuál /es no. Nota: el campamentista intenta pedir cada elemento una vez.
El código funciona correctamente? Si considera que no funciona correctamente detalle el/ los error /es que considera que tiene.

```
Campamentista[J:1..M]
 numElem:= 1;
 obtenidos[1..12]:= false;

while (numElem<=12) do
  send empezar [J] ();
  send quiero (J, numElem);
  listo:= false;
  while (not listo)
     if (not empty fin[J]) then
          receive fin[J] ();
          obtenidos[numElem]:= false;
          listo:= true;
     □
      if (not empty toma[J]) then
          receive toma[J] ();
          obtenidos[numElem]:= true;
          listo:= true;
   end;
  numElem:= numElem + 1;
end;
send meVoy[J]();

<Imprime el arreglo que informa que elementos obtuvo >
```

```
Encargado::
 while (true) do
      receive quiero (J, elem);
      < busca el elemento elem>
      send toma[J]();
end;
```

```
Reloj[K:1..M];
 final:= false;
 while (not final)
   if (not empty empezar[K]) then
       receive empezar [K] ()
       delay (7);
       send fin [K] ();
   □
   if (not empty meVoy[K]) then
      receive meVoy[K] ()
      final:= true;
end;
```

2- M personas concurren a un gimnasio para hacer su rutina diaria. Cada persona cuando llega al gimnasio da su nombre, su bolso con ropa para bañarse y su portafolio o cartera personal (dependiendo de si es hombre o mujer) a la secretaria. La secretaria debe verificar que la persona tenga la cuota paga, si es así, la deja ingresar avisándole que puede guardar sus pertenencias en un locker.
Una vez que la persona ingresó al gimnasio pide que alguno de los J profesores la atienda (el profesor que se le debe asignar a la persona es aquel que menos personas esté atendiendo en ese momento). El profesor que la atiende toma su rutina y le indica cuál tipo de máquina debe utilizar. La persona busca una máquina del tipo (la que hace más tiempo que no se utiliza) y si en ese momento la máquina no está disponible, le pide a su profesor que le diga otro tipo de máquina a utilizar. La persona se retira del gimnasio cuando termina su rutina o cuando 5 máquinas no estuvieron disponibles, en cualquiera de los casos debe retirar sus pertenencias personales.

Notas: La persona no conoce el número de su locker particular.
Existen N máquinas de cada tipo. La persona no elige con que profesor hace su rutina, una vez que consigue uno trabaja con ese hasta que se retira del gimnasio y tampoco elige cuál máquina de cada tipo utiliza.
Cada persona utiliza cada máquina durante 5 minutos.
Un profesor puede atender a 0, 1 ó más personas.
Suponga que cada máquina dispone de una función que le retorna cuanto tiempo hace que está disponible.
Maximice la concurrencia para TODAS las taréas. Modelice en ADA.

PROGRAMACION CONCURRENTE

PRIMER PARCIAL

1. Suponga un juego donde hay 30 competidores. Cuando los jugadores llegan avisan al encargado, **una vez que están los 30** el encargado del juego les entrega un numero aleatorio del 1 al 15 de tal manera que dos competidores tendrán el mismo numero. (Suponga que existe una función DarNumero() que devuelve en forma aleatoria un numero del 1 al 15, **el encargado no guarda el numero que les asigna a los competidores**). **Una vez que ya se entregaron** los 30 números, los competidores buscaran **concurrentemente** su compañero que tenga el mismo numero (tenga en cuenta que pueden empezar a buscar cuando todos los competidores tengan el numero **no antes**; Además la búsqueda de un jugador no interfiere con la búsqueda de otros que tengan distinto numero). Cuando los competidores se encuentran permanecen en una sala durante 15 minutos y dejan de jugar. Luego cada uno de los competidores avisa al encargado que termino de jugar y espera a que su compañero (el que tenia el mismo numero) también avise que finalizo para luego irse ambos; el encargado cuando llega el segundo competidor les devuelve a ambos el resultado que obtuvieron que es el orden en que se van. (Los primeros en irse, tendrán como resultado 1, los últimos 15). Para modelizar el tiempo utilice la función Delay(x) que produce un retardo de x minutos.

Modelice Con Semáforos

2. Se tienen N procesos que comparten el uso de una CPU.

Un proceso, cuando necesita utilizar la CPU, le pide el uso, le proporciona el trabajo que va a ejecutar y el tiempo que insume ejecutar el trabajo. Luego, el proceso se queda dormido hasta que la CPU haya finalizado de ejecutar su trabajo, momento en que es despertado y prosigue su ejecución normal.

La CPU funciona de la siguiente manera. Cada vez que el proceso i (i:1..N) pide CPU esta lo pone en el lugar isesimo de una lista de procesos esperando para ejecutar. La CPU recorre circularmente la lista de procesos esperando a ser ejecutados (del 1 al N) , si el proceso esta listo lo saca de la lista y lo ejecuta durante un lapso de 10 mls (o un lapso menor, si el tiempo de ejecución del proceso es menor a 10 mls). Una vez que ejecutó un proceso, si todavía le queda al proceso tiempo de ejecución lo vuelve a poner en su lugar en la lista y pasa al siguiente. Si el proceso terminó de ejecutar su trabajo, la CPU lo despierta para que continúe su ejecución y continúa con el siguiente proceso de la lista. Cuando la lista está vacía, la CPU no debe hacer nada, simplemente debe esperar a que algún proceso ingrese a la lista de procesos en espera de ejecución, para empezar a trabajar.

Tenga en cuenta que existe la función delay(x) que retarda un proceso durante x mls. La ejecución de un procesó puede ser modelizada utilizando esta función.

No haga suposiciones acerca de los tiempos de ejecución de los procesos

Modelice utilizando Monitores

PROGRAMACION CONCURRENTE

SEGUNDO RECUPERATORIO

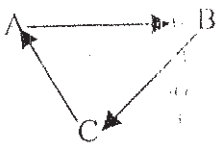
En una carpintería se realizan muebles ensamblando 3 partes, existen N1 carpinteros para realizar la parte1, N2 para realizar la parte 2, N3 para la parte 3 mas N carpinteros que se encargan de ensamblar las 3 partes del mueble.
Los encargados de ensamblar deben tomar una pieza de cada clase juntar las partes y **una vez ensambladas** los carpinteros que le dieron la pieza pueden seguir trabajando(no antes)
En la carpintería solo se armaran 30 muebles y no se podrán producir piezas de mas.
El armado debe realizarce en forma **concurrente** solo pueden esperarce entre los carpinteros de un mismo tipo cuando terminan una pieza hasta que los tome un ensamblador (es decir puede pensar que los ensambladores toman las piezas de a uno)
Todos los procesos deben terminar correctamente no pueden quedar procesos colgados.
Modelice con Semáforos.

Encuentre la intersección entre 4 conjuntos en forma **concurrente**, teniendo en cuenta que existen 4 procesos que tienen localmente cada conjunto, es decir el proceso uno el conjunto uno, así hasta el proceso cuatro el conjunto cuatro . No debe suponer que la inicialización de alguna región contenga los valores de los conjuntos.
Modelice con Regiones Criticas Condicionales.

PROGRAMACIÓN CONCURRENTE

PRIMER RECUPERATORIO

1. Suponga que existen N barcos que recorren un canal llevando gente. Su recorrido es el siguiente:



A-B, B-C, C-A.
Los barcos parten desde A, van hasta B y luego van hasta C, y de C nuevamente vuelven a A. Luego de hacer 7 vueltas completas paran durante 1h a descansar y vuelven a salir.
No puede haber mas de tres barcos haciendo el mismo recorrido simultáneamente.
Cuando un barco llega a un punto de parada, los pasajeros que quieren bajar lo hacen. Una vez que bajaron todos el barco se fija si puede partir (teniendo en cuenta que no puede haber mas de tres barcos haciendo el mismo recorrido), si no puede se queda esperando sin subir gente hasta poder partir.
Cuando se dan las condiciones de poder seguir permite subir gente durante 5' o hasta que se llene su capacidad. Suponga que cada barco tiene capacidad para 30 pasajeros.
Existen R pasajeros que conocen la parada en la que suben y en cual bajan. Cuando un pasajero llega a su punto de partida trata de subir en el primer barco que parta, cuando su barco llega a la parada destino se baja. Tenga en cuenta que el barco no conoce el destino de sus pasajeros.
Suponga que como mínimo cada recorrido del barco tarda 20'. Maximice la concurrencia.
Modelice con Pasajes de Mensajes Asincrónicos o Sincrónicos según crea conveniente.

2. Suponga un juego donde hay 30 competidores. Cuando los jugadores llegan avisan al encargado, **una vez que están los 30** el encargado del juego les entrega un numero aleatorio del 1 al 15 de tal manera que dos competidores tendrán el mismo numero. (Suponga que existe una función DarNumero() que devuelve en forma aleatoria un numero del 1 al 15, **el encargado no guarda el numero que les asigna a los competidores**). **Una vez que ya se entregaron** los 30 números, los competidores buscaran **concurrentemente** su compañero que tenga el mismo numero (tenga en cuenta que pueden empezar a buscar cuando todos los competidores tengan el numero **no antes**; Además la búsqueda de un jugador no interfiere con la búsqueda de otros que tengan distinto numero). Cuando los competidores se encuentran permanecen en una sala durante 15 minutos y dejan de jugar. Luego cada uno de los competidores avisa al encargado que termino de jugar y espera a que su compañero (el que tenía el mismo numero) también avise que finalizo para luego irse ambos, el encargado cuando llega el segundo competidor les devuelve a ambos el resultado que obtuvieron que es el orden en que se van (Los primeros en irse, tendrán como resultado 1, los últimos 15). Para modelizar el tiempo utilice la función Delay(x) que produce un retardo de x minutos.
Modelice Con Regiones Criticas.

6. Describa en qué consiste el problema de los "Drinking Philosophers"

El problema de los "Drinking Philosophers", planteado y resuelto por Chandy y Misra, es una generalización del problema de los "Dining Philosophers" que captura la esencia de los problemas de resolución de conflictos en sistemas distribuidos.

El problema consiste en lo siguiente:

- Varios procesos, llamados filósofos, son colocados en los nodos (vértices) de un grafo no dirigido finito G, con un filósofo en cada nodo.
- Un filósofo está en uno de 3 estados: *tranquilo*, *sediento* o *bebiendo*.
- Asociado con cada arista en G hay una botella. Un filósofo puede beber sólo de botellas asociadas con sus aristas incidentes.
- Un filósofo *tranquilo* podría pasar al estado *sediento*.
- Un filósofo *sediento* necesita un conjunto no-vacío de botellas de las que quiere beber y puede necesitar conjuntos diferentes de botellas para diferentes sesiones.
- Cuando posee todas las botellas que necesita, un filósofo sediento comienza a beber; un filósofo sediento permanece sediento hasta que obtiene todas las botellas que necesita para beber.
- Al entrar al estado de "*bebiendo*" un filósofo permanece en ese estado por un periodo finito, después del cual pasa al estado "*tranquilo*".
- Un filósofo *tranquilo* podría permanecer en ese estado por un periodo de tiempo arbitrario.
- Dos filósofos son vecinos si y solo si hay una arista entre ellos en G.
- Filósofos vecinos pueden enviarse mensajes entre sí. Estos mensajes son enviados en un tiempo arbitrario, pero finito.
- Los recursos, como las botellas, también son codificados transmitidos como mensajes.

La solución al problema de los "Drinking Philosophers" debe ser no-probabilística, y además debe satisfacer las siguientes restricciones:

Fairness: Ningún filósofo debe permanecer por siempre sediento.

Simetría: Todos los filósofos obedecen precisamente las mismas reglas para adquirir y liberar botellas. No hay prioridades ni otras formas de ordenamiento estático parcial especificados externamente entre los filósofos o las botellas.

Economía: Un filósofo envía y recibe un número finito de mensajes entre las transiciones de estado. En particular, los filósofos permanentemente tranquilos no envían o reciben un número finito de mensajes.

Concurrencia: La solución no niega la posibilidad de diferentes filósofos bebiendo simultáneamente de diferentes botellas.

Limitación: El número de mensajes en tránsito, en cualquier momento, entre cualquier par de filósofos es limitado. Más aún, el tamaño de cada mensaje es limitado.

Plantee al menos un esquema de solución

Se plantea un esquema de solución distribuida del problema de los "Drinking Philosophers":

En el nodo n_i del grafo G, el conjunto de botellas compartidas sobre la arista (n_i, n_j) se denota por `B[j,i]`, y es equivalente a `B[i,j]`.

Para todo n_j perteneciente al conjunto de nodos adyacentes a n_i (Ady_i), y para cada botella b_k perteneciente a `B[j,i]`, el nodo n_i emplea la variable `boolean necesita_botella[j,k,i]` inicializada en `false`, para indicar si se necesita la botella b_k perteneciente a `B[j,i]`.

La necesidad de acceder a recursos compartidos se indica en el nodo n_i por la variable `boolean sediento`$_i$, inicializada en `false`. Para poder acceder a los recursos compartidos que necesita, n_i debe adquirir todas las botellas correspondientes a ese recurso. Por ejemplo, b_k pertenece a `B[j,i]`, de manera tal que `necesita_botella[j,k,i] = true` para todo n_j perteneciente a Ady_i.

Algunas variables adicionales empleadas por el nodo n_i son, para todo n_j perteneciente a Ady_i, la variable `boolean tiene_turno[j,i]`, y para todas las botellas b_k las variables `boolean tiene_botella[j,k,i]`, empleada para indicar si n_i tiene la botella b_k y `boolean debe_botella[j,k,i]`, la cual es inicializada en `false` y se utiliza para indicar si n_i ha pospuesto el envío de b_k a n_j hasta que haya terminado de acceder a los recursos compartidos. Las variables `tiene_turno` y `tiene_botella` deben ser inicializadas consistentemente a través de las aristas. Además, la variable `tiene_turno` debe ser inicializada para inducir una orientación acíclica en G.

Los nodos n_i también utilizan variables auxiliares de conjunto X_i e Y_i, las cuales se inicializan con el valor `VACIO`. Por ejemplo, un conjunto X de botellas se envía mediante `botella(X)`.

Finalmente, se utiliza `turn` para enviar un turno a través de una arista.

1) K cobras desean entrar a una granja a comer. Cuando llegan se comunican con un encargado para que éste les asigne una pareja. Cuando una cebra tiene asignada una pareja entra al salón y come. Para salir de la granja debe sincronizarse con su pareja. Esta sincronización debe realizarse entre las dos cebras directamente, sin ningún intermediario. Dado el siguiente código: usted piensa que funciona correctamente? Justifique. Si la respuesta es negativa, corrija lo necesario del código.

```
(en el programa principal se declara el arreglo de tareas Cebra y se les asigna el identificador)
Var Cebras[1:K] of Cebra
        for i:=1..K do Cebras[i].ide(i);
Task Encargado is
        entry darCebraPareja (in c1:cebra; out c2: cebra)
Task Body Encargado is
begin
        for i:= 1 .. K/2 loop
            accept darCebraPareja (in p1,out pareja1) do
                accept darCebraPareja (in p2,out pareja2) do
                                pareja1 := p2; pareja2 := p1
                end;
            end;
        end loop;
end;
Task Type Cebra is
        entry ide(i:integer)
        entry mevoy()
Task Body Cebra is
begin
        accept ide (in i:integer) do
                yo:=i;
        Encargado.darCebraPareja( yo, miPareja);
        <Coma>
        Select
            accept mevoy()
        else
            Cebras[miPareja]. mevoy ()
End;
```

2) En una casa de comidas rápidas trabajan N despachadores. A la misma llegan M clientes, donde cada uno espera ser atendido por cualquiera de los despachadores. Una vez que el cliente es atendido entrega su pedido al despachador, el cual se dirige al cocinero, le da el pedido y espera la comida realizada. Luego el despachador le da al cliente un ticket que contiene el monto a pagar. El cliente se dirige a la caja, paga su ticket recibe la factura y con ella se dirige al despachador que lo atendió, le entrega la factura y obtiene el pedido. Luego de tomar el pedido el cliente se retira y el despachador puede seguir atendiendo. Resuelva utilizando semáforos. Maximice la concurrencia.
Nota: los clientes deben ser atendidos en orden de llegada, y si un cliente permanece más de 25 minutos esperando que lo atiendan se retira de la casa de comidas. Siempre hay comida para satisfacer cualquier pedido. Modelice todo lo enunciado.

Programación Concurrente **1er Fecha - Completo** **Tema 2**

1. En un parque de diversiones se realiza M entregas de pases para juegos donde en cada entrega se otorgan 7 pases para los juegos. El encargado de entregar los pases es el jefe del parque, el cual entrega los 7 pases a la persona que mas pedidos hasta el momento. Una vez que la persona recibe los pases le indica al jefe en que juego/s va a utilizar cada pase. Luego de recibir los nombres de los juegos el jefe realiza una actividad por 15 minutos para luego volver a repartir. Modelice en Ada. Maximice la concurrencia.
2. Se desea modelar el funcionamiento de un banco el cual se encarga de cobrar únicamente el servicio de seguro de sus clientes, en el cual existen 5 cajas donde se cobra.
   Existen P personas que desean pagar su seguro en el banco. Para esto cada una selecciona la caja donde hay menos personas esperando, una vez seleccionada espera a ser atendido según el orden de llegada. Cuando lo atienden, si esperó más de 20 minutos entonces el banco le regala el cobro del servicio, en caso contrario, debe abonar una cierta cantidad dependiendo de la categoría de cliente (en caso de no pagar justo el empleado debe darle el vuelto).
   Si la persona esperó más de 20 minutos, puede optar por levantar una queja. En ese caso, una vez que se retiro de la caja (sin pagar), se dirige al departamento de quejas donde un supervisor toma los datos de la persona (DNI de la persona, monto) y el número de caja que lo atendió. Cuando se han levantado más de 15 quejas para una misma caja el supervisor cierra la caja, dejando pasar gratis a todas las personas que estaban esperando en ella.
   Implemente utilizando pasaje de mensajes sincrónico o asincrónico según crea conveniente. Aclaraciones:
   - Existe una función Costo que retorna la cantidad que debe pagar quien la invoca.
   - Suponer que nunca se necesitará cerrar todas las cajas.
   - Existe una función que dado el dni de la persona devuelve si la misma quiere o no realizar una queja.
   - MAXIMICE LA CONCURRENCIA.

PROGRAMACION CONCURRENTE

SEGUNDO RECUPERATORIO

En una carpintería se realizan muebles ensamblando 3 partes, existen N1 carpinteros para realizar la parte1, N2 para realizar la parte 2, N3 para la parte 3 mas N carpinteros que se encargan de ensamblar las 3 partes del mueble.
Los encargados de ensamblar deben tomar una pieza de cada clase juntar las partes y **una vez ensambladas** los carpinteros que le dieron la pieza pueden seguir trabajando(no antes)
En la carpintería solo se armaran 30 muebles y **no** se podrán producir piezas de mas.
El armado debe realizarce en forma **concurrente** solo pueden esperarce entre los carpinteros de un mismo tipo cuando terminan una pieza hasta que los tome un ensamblador (es decir puede pensar que los ensambladores toman las piezas de a uno)
Todos los procesos deben terminar correctamente no pueden quedar procesos colgados.
Modelice con Semáforos.

Encuentre la intersección entre 4 conjuntos en forma **concurrente**, teniendo en cuenta que existen 4 procesos que tienen localmente cada conjunto, es decir el proceso uno el conjunto uno, así hasta el proceso cuatro el conjunto cuatro . No debe suponer que la inicialización de alguna región contenga los valores de los conjuntos.
Modelice con Regiones Criticas Condicionales.

1-Hay N personas que deben hacer un tramite en el banco. Para esto primero hacen cola frente al mostrador principal. Detrás del mostrador hay R empleados que alienden a las personas y según el tramite que deben hacer les asignan una de las dos cajas disponibles en el banco y un número (suponer una funcion "evaluarTramite" que dados los datos del cliente devuelve una caja y un número de tramite). **La evaluacion de tramites por parte de diferentes empleados debe ser concurrente.**
Luego de ser atendidas, las personas se paran frente a la caja a la que fueron asignadas (sin hacer cola) y esperan a ser llamadas por el cajero de dicha caja según su numero (el numero asignado a las personas esta entre 1 y k).
Cuando en el mostrador central se juntan mas de cinco tramites para alguna caja, el cadete del banco lleva los tramites al cajero de la caja correspondiente.
Cada cajero aliende los tramites asignados a su caja. Luego de atender el tramite de una persona, la llama por el numero de tramite y le da un ticket. Cuando la persona toma el ticket el cajero aliende el siguiente tramite. La persona que fue atendida se va del banco.
Cuando un cajero no tiene mas tramites, espera hasta que el cadete le lleve nuevos tramites.
Nota: tener en cuenta que cuando el cajero termina de atender un trámite y llama a una persona por el número la persona puede no estar todavía frente a la caja (la persona puede tardar mas tiempo en ir desde el mostrador a la caja, que el cajero en atender su trámite).
Resolver con monitores.

2-Una central que recibe resultados de experimentos de dos tipos de investigadores. Normalmente el valor de los experimentos varía entre -5 y 100.
Si el valor obtenido de algún resultado de experimento es menor al limite inferior (-5) entonces la central tomará los N resultados de experimentos siguientes del mismo tipo de investigador (N es un número random entre 1..7), y los mandará a almacenar. El "almacenar" un resulatado no debe interferir la recepción de los N-1 restantes resultados.
Si el valor obtenido de algún resultado de experimento supera el limite superior (100), la central esperará el siguiente resultado de experimento del mismo tipo de investigador durante, a lo sumo, 2 minutos y luego seguirá normalmente.
Si la central no se recibe ningún resultado de experimento durante 20 minutos, se considera que los investigadores no realizan más experimentos y finaliza su ejecución.
Los investigadores piden los datos para realizar el experimento a una tarea administrador dependiendo el tipo del investigador. Para esto hay S tareas de tipo administrador 1 y Q tareas del tipo administrador 2.
Una vez recibidos los datos realizan el experimento y esperan durante 5 segundos que la central la tome el resultado, en caso de no ser recibido por la central, el experimento se desecha y se genera uno nuevo con otros datos. Si la central no toma 6 resultados de experimentos consecutivos, se considera que no está funcionando y finaliza la ejecución del investigador.
Nota: no deben quedar tareas colgadas.
Modelice en Ada.

llegan L alumnos que comparten la atención del empleado de la oficina de alumnos.
Alumno, cuando necesita la atención del empleado, le pide atención, le proporciona el trámite que necesita realizar y tiempo que lnaumo realizarlo. Luego, el alumno se queda dormido hasta que el empleado haya finalizado de realizar su trámite, momento en que es despertado y prosigue su ejecución normal.
El empleado de la oficina atiende de la siguiente manera. Cada vez que el alumno I (I: 1..L) pide atención anota lo pone en el lugar lesimo de una lista de alumnos esperando para atender. El empleado recorre circularmente la lista de alumnos esperando a ser atendidos (del 1 al L) , si el alumno esta listo lo saca de la lista y lo ejecuta durante un lapso de 5 min (o un lapso menor, si el tiempo de atención es menor a 5 min). Una vez que atendió un alumno, si todavía le queda al alumno tiempo de atención lo vuelvo a poner en su lugar en la lista y pasa al siguiente. Si el alumno terminó su trámite, el empleado lo despierta para que continúe su ejecución y continúa con el siguiente alumno de la lista. Cuando la lista de trabajos está vacía, el empleado no debe hacer nada, simplemente debe esperar a que algún alumno ingrese a la lista de trabajos en espera de ejecución, para empezar a trabajar.
No haga suposiciones acerca de los tiempos de ejecución de las tareas de los alumnos.
Modelice utilizando Monitores.

2- Suponga que se tienen X empleados los cuales deben realizar exactamente L productos en total. Cada empleado trabaja de manera independiente. El siguiente código cumple lo pedido?. Si su respuesta es afirmativa JUSTIFIQUE y en caso de considerar que no resuelve lo pedido JUSTIFIQUE indique el error y modifique el código.

```
Var
 cant: integer:= 0;   sem: semaforo:=1;

Empleado[i:1..X]::
 Var
  c: integer;
 Begin
  P(sem);
  c:= cant;
  V(sem);
 While (c < L) loop
   P(sem);
   cant:= cant+1;
  V(sem);
   <fabrica un producto>
   P(sem);
   c:= cant;
  V(sem);
 end loop;
End;
```

4. Se tiene una tabla distribuida entre 10 procesos (cada proceso tiene 10000 registros). La tabla contiene el nombre y la dirección de cada persona. Se debe encontrar la dirección de "Juan Pérez" (seguro está en la tabla, y sólo una vez). Ni bien se encuentra el dato todos los procesos deben dejar de buscar. ***Resolverlo con Pasaje de Mensaje Asincrónico utilizando solo un canal.***

5. En un entrenamiento de futbol hay 20 jugadores que forman 4 equipos *(cada jugador conoce el equipo al cual pertenece llamando a la función **DarEquipo()**)*. Cuando un equipo está listo *(han llegado los 5 jugadores que lo componen)*, debe enfrentarse a otro equipo que también esté listo (*los dos primeros equipos en juntarse juegan en la cancha 1, y los otros dos equipos juegan en la cancha 2)*. Una vez que el equipo conoce la cancha en la que juega, sus jugadores se dirigen a ella. Cuando los 10 jugadores del partido llegaron a la cancha comienza el partido, juegan durante 50 minutos, y al terminar todos los jugadores del partido se retiran *(no es necesario que se esperen para salir)*. ***Resolver con monitores.***

6. **SOLO PARA LOS QUE RINDEN COMPLETO.** Se desea administrar el acceso de *N* usuarios a una Base de Datos. Solo se le puede permitir el acceso a usuario a la vez. Cada usuario espera a lo sumo 10 minutos a que le den el acceso, pasado ese tiempo se retira sin acceder a la base. ¿El siguiente código funciona correctamente? ***JUSTIFICAR la respuesta.***

```
Procedure Ejercicio3 is

  Task Administrador is
     entry Pedir (id: integer);
     entry Dejar;
  end administrador;

  Task type Usuario is
     entry identificacion (id: integer);
     entry usar;
     entry tiempo;
  end usuario;

  Task type Reloj is
     entry comenzar(id: integer);
  end reloj;

  v_usuarios: array (1..N) of usuario;
  v_relojes: array (1..N) of reloj;

  Task body Usuario is
     aux: integer;
  begin
     accept identificacion (id: integer) do
        aux:= id;
     end identificacion;
     administrador.pedir(aux);
     v_relojes(aux).comenzar(aux);
     select
        accept tiempo;
     or
        accept usar;
        (Usa la Base de Datos)
        administrador.dejar;
     end select;
  end usuario;

  Task body Reloj is
     aux: integer;
  begin
     accept comenzar (id: integer) do
        aux:= id;
     end comenzar;
     delay(600.0);
     v_usuarios(aux).tiempo;
  end reloj;

  Task body Administrador is
     aux: integer;  cola: queue;
  begin
     loop
        accept Pedir (id: integer) do
           aux := id;
        end Pedir;
        if (libre) then
           v_usuarios(aux).usar;
           libre :=false;
        else  push (cola, aux);
        end if;
     or
        accept dejar;
        if (not empty(cola)) then
           pop(cola, aux);
           v_usuarios(aux).usar;
        else libre := true;
        end if;
     end loop;
  end Administrador;

Begin
  for i in 1..N loop
    v_usuarios(i).identificacion(i);
  end loop;
End Ejercicio3;
```

1) Suponga que existen N usuarios que deben ejecutar su programa, para esto comparten K procesadores.

Los usuarios solicitan un procesador al administrador. Una vez que el administrador les entregó el número de procesador, el usuario le da su programa al procesador que le fue asignado. Luego el usuario espera a que:

- El procesador le avise si hubo algún error en una línea de código con lo cual el usuario arregla el programa y se lo vuelve a entregar al procesador, es decir queda nuevamente en la cola de programas a ejecutar por su procesador. El usuario no termina hasta que el procesador haya ejecutado su programa correctamente(sin errores).

- El procesador le avise que su programa termino, con lo cual termina su ejecución.

El administrador tomará los pedidos de procesador hechos por los usuarios y balanceara la carga de programas que tiene cada procesador, de esta forma le entregará al usuario un número de procesador.

El procesador ejecutará un Round-Robin de los programas listos a ejecutar. Cada programa es ejecutado línea por línea por medio de la funcion EJECUCIÓN la cual devuelve:

1 error en la ejecución.
2 normal.
3 fin de programa.

Nota: Suponga que existe también la función LineaSiguiente que dado un programa devuelve la línea a ser ejecutada. Modelice con Ada. **Maximice la concurrencia en la solución.**

2) En un centro cultural se está realizando una exposición de pintura. La misma se lleva a cabo dentro de la sala "Da Vinci", además existen X personas que quieren entrar en la sala.
Al entrar a la sala las personas permanecen D minutos mirando las pinturas y luego se retiran. La sala permite a lo suma J personas (J < X) simultáneamente dentro de la misma. Si la sala está llena una persona que solicita la entrada queda demorada. Cuando se desocupa un lugar porque alguien se va de la sala, la siguiente persona en entrar debe ser aquella que esté esperando desde hace más tiempo.
Se desea resolver el problema usando monitores. Dada la solución presentada a continuación determinar si la misma respeta el enunciado. En caso de no ser así justifique y modifique lo necesario en el monitor para que lo respete.

No es correcto

```
Persona [i:1..X]:
   Sala.pasar();
   delay(D);
   Sala.salir();
```

```
Monitor Sala;
 Var  cant: integer:= 0; c: condition;

 procedure pasar();
 begin
   if cant = J → WAIT (c);
   cant = cant + 1;
  end;

 procedure salir();
 begin
   cant = cant – 1;
   SIGNAL (c);
 end;
End;
```

1- Un banco abre sus puertas a las 10:00 hs (suponga que existe una señal externa que avisa). El banco tiene 2 puertas, y cada una posee un detector de metales.
Existen además M clientes. Los clientes llegan al banco o intentan entrar por una de las puertas(suponga que el cliente conoce la puerta). Al estar en frente de una puerta, se activa el detector. Si en 5 segundos no lo dejan pasar, la persona se va. Además, los clientes para salir del banco, sólo deben esperar que la puerta esté libre. Los clientes conocen la puerta por la cuál entran y por la cuál salen. La entrada y la salida por la puerta deben ser modelizadas como un retardo de 2 segundos realizado por los clientes. El banco permanece abierto durante 5 hs, a partir del momento en que abrió sus puertas. Una vez pasadas las 5 hs. no deja entrar más clientes y espera a que salgan todos los que están dentro para poder cerrar sus puertas.
Una vez que un cliente entra al banco deja en un único mostrador una carpeta con los datos de su trámite y queda esperando que lo llamen para devolverle la carpeta una vez atendido. En el banco existen E empleados que atienden los trámites de los clientes. Los empleados retiran las carpetas del mostrador, procesan el trámite y una vez terminado llaman al cliente y le devuelven la carpeta. Una vez que el cliente recibe su carpeta procede a salir del banco y el empleado continua su tarea.
Modelice en ADA.

2- Suponga que existe una bolsa con X caramelos y tenemos N niños que quieren comer los caramelos hasta que se acaben. El siguiente monitor soluciona lo pedido?. JUSTIFIQUE su respuesta. Si considera que no soluciona lo pedido.

```
Monitor Caramelos
Procedure quieroComer (ldo:integer; var ok:boolean);
Begin
If (cantidadComida < X ) then ok:= true
Else ok:= false
End;
Procedure come;
Begin
cantidadComida:= cantidadComida + 1;
End;
```

```
Procedure Niño[I:1..N]::
Var puedo:bolean;
Begin
puedo:= true
while (puedo)
begin
  Caramelos.quieroComer(I,puedo)
  If (puedo) then Caramelos.come
end
```

...llanan L alumnos que comparten la atención del empleado de la oficina de alumnos.
El alumno, cuando necesita la atención del empleado, le pide atención, le proporciona el trámite que necesita realizar y el tiempo que insumo realizarlo. Luego, el alumno se queda dormido hasta que el empleado haya finalizado de realizar su trámite, momento en que es despertado y prosigue su ejecución normal.
El empleado de la oficina atiende de la siguiente manera. Cada vez que el alumno I (I: 1..L) pide atención este lo pone en el lugar isésimo de una lista de alumnos esperando para atender. El empleado recorre circularmente la lista de alumnos esperando a ser atendidos (del 1 al L) , si el alumno esta listo lo saca de la lista y lo ejecuta durante un lapso de 5 mls (o un lapso menor, si el tiempo de atención es menor a 5 mls). Una vez que atendió un alumno, si todavía le queda al alumno tiempo de atención lo vuelve a poner en su lugar en la lista y pasa al siguiente. Si el alumno terminó su trámite, el empleado lo despierta para que continúe su ejecución y continúa con el siguiente alumno de la lista. Cuando la lista de trabajos está vacía, el empleado no debe hacer nada, simplemente debe esperar a que algún alumno ingrese a la lista de trabajos en espera de ejecución, para empezar a trabajar.
No haga suposiciones acerca de los tiempos de ejecución de las tareas de los alumnos.
Modelice utilizando Monitores.

2- Suponga que se llenan X empleados los cuales deben realizar exactamente L productos en total. Cada empleado trabaja de manera independiente. El siguiente código respeta lo pedido?. Si su respuesta es afirmativa JUSTIFIQUE, y en caso de considerar que no resuelve lo pedido JUSTIFIQUE indique el error y modifique el código.

```
Var
 cant: Integer:= 0;   sem: semaforo:=1;

Empleado[i:1..X]::
 Var
  c: Integer;
 Begin
 ( P(sem);
 | c:= cant;
 | V(sem);
 While (c < L) loop
 ( P(sem);
 | cant:= cant+1;
 V(sem);
  <fabrica un producto>
 ( P(sem);
 | c:= cant;
 | V(sem);
 end loop;
End;
```

1- En un galpón hay M tipos de materiales dispuestos en cajones que deben cargarse en M camiones, uno para cada tipo de material. Los cajones del material 1 deben cargarse en el camión 1, y así del 2 hasta M.
Hay M clases de empleados, los del tipo1 son los que van a encargarse del cargamento 1, y así los del tipo M van a ocuparse del cargamento M.
Para cada clase de cargamento con (I:1..M) hay 50 empleados.
Los empleados deben entrar al galpón y tomar un cajón del material del tipo que les corresponde y llevarlo al camión.
Cargando el camión solo puede haber 1 empleado a la vez.
En caso de que no queden mas cajones de su tipo dejaría de trabajar y se iría. Existen X cajones de cada tipo de material.
El empleado que carga el ultimo cajón en el camión de su tipo, le avisa al camionero para que arranque el camión y se vaya.
Para entrar al galpón hay que pasar por un pasillo en el cual los empleados entran y salen de a uno. No puede haber más de un empleado en el pasillo al mismo tiempo. Siempre la prioridad para cruzar el pasillo la tienen los empleados que están con su cajón en la mano, o sea los que salen tienen prioridad por sobre los que quieren entrar.
Si hay muchos que quieren salir la prioridad es para el que esta primero en la cola, lo mismo para los que están por entrar.
Modelice con monitores.

2- Dado el siguiente código en Ada, ¿Puede haber deadlock?. Si su respuesta es afirmativa indique donde se produce el deadlock y modifique el algoritmo para solucionarlo. Si su respuesta es negativa explique cómo se asegura la finalización de todas las tareas.

```
Task Type Consumidor is
   Entry empezar (Ido: Integer);
   Entry finTiempo;
   Entry señal;
End Empleado

Consumidores:array(1..15) of Empleado

Task Body Consumidor is
Myido: Integer;

Begin
Accept empezar (Ido: Integer) do
   MyIde:= Ido;
End empezar;

ControladoresDeTiempo[myIde].empezar(myIde)

Select
 accept señal;
 controladoresDeTiempo [myIde].meFui;
or
 accept finTiempo;
End select;
End Consumidor;
```

```
Task Type ControladorDeTiempo is
   entry empezar (Ido: Integer);
   entry meFui;
End ControladorDeTiempo

Controladores:array(1..15) of
ControladorDeTiempo

Task Body ControladorDeTiempo is
MyIde: Integer;
B: boolean:= true;
Begin
  Accept empezar (Ido: Integer) do
    myIde:= Ido;
  end empezar;

Select
  Accept meFui;
Or
  delay (20)
  while (b) loop
    select
      consumidores[myIde].finTiempo
      b:= false;
    else
      select
        accept meFui;
        b:= false;
      else
        null;
      end select;
    end select;
  end loop;
End ControladorDeTiempo;
```

```
Task Administrador;

Task Body Administrador is
emple : integer;
Begin
  consu:= random(1,15);
  select
    consumidores[consu].señal;
  or
      delay (10);
      put ("Me voy");
  end select;
End Administrador;
```