Resumen 1er Parcial

Tipos de ejercicios de semáforos

Productor/consumidor

Para identificar estos ejercicios me tengo que fijar que haya una capacidad limitada para agregar recursos (capacidad N contenedores)
Se deben llevar enteros “libre” y “ocupado” para que recorran circularmente.

Enunciados de ejemplo:

- Ejercicio 5 de la práctica
  En una empresa de logística de paquetes existe una sala de contenedores donde se preparan las entregas. Cada contenedor puede almacenar un paquete y la sala cuenta con capacidad para N contenedores. Resuelva considerando las siguientes situaciones:
  a) La empresa cuenta con 2 empleados: un empleado Preparador que se ocupa de preparar los paquetes y dejarlos en los contenedores; un empelado Entregador que se ocupa de tomar los paquetes de los contenedores y realizar la entregas. Tanto el Preparador como el Entregador trabajan de a un paquete por vez.
  b) Modifique la solución a) para el caso en que haya P empleados Preparadores.
  c) Modifique la solución a) para el caso en que haya E empleados Entregadores.
  d) Modifique la solución a) para el caso en que haya P empleados Preparadores y E empleadores Entregadores

- Ejercicio 9 de la práctica

Resolver el funcionamiento en una fábrica de ventanas con 7 empleados (4 carpinteros, 1 vidriero y 2 armadores) que trabajan de la siguiente manera:
• Los carpinteros continuamente hacen marcos (cada marco es armando por un único
carpintero) y los deja en un depósito con capacidad de almacenar 30 marcos.
• El vidriero continuamente hace vidrios y los deja en otro depósito con capacidad para 50 vidrios.
• Los armadores continuamente toman un marco y un vidrio (en ese orden) de los depósitos correspondientes y arman la ventana (cada ventana es armada por un único armador).

- Ejercicio de parcial recuperatorio
  Resolver con SEMÁFOROS el siguiente problema. En un restorán trabajan C cocineros y M mozos. De forma repetida, los cocineros preparan un plato y lo dejan listo en la bandeja de platos terminados, mientras que los mozos toman los platos de esta bandeja para repartirlos entre los comensales. Tanto los cocineros como los mozos trabajan de a un plato por vez. Modele el funcionamiento del restorán considerando que la bandeja de platos listos puede almacenar hasta P platos. No es necesario modelar a los comensales ni que los procesos terminen.
  NOTA: es el problema de productores/consumidores con tamaño de buffer limitado visto en la página 14
  de la teoría 4.

Ejercicio resuelto

(ejercicio parcial recuperatorio)

```
buffer platos[P]
int ocupado = 0
int libre = 0
Sem vacio = P
Sem lleno = 0
Process Cocineros[id:1..C]{
    while (true){
        plato = preparar()
        P(vacio) //me fijo que haya lugar
        P(mutex)
        platos[libre] = plato
        libre = (libre+1) mod P
        V(mutex)
        V(lleno) //asviso que hay plato
    }
}
Process Mozos[id:1..M]{
    while (true){
        P(lleno)
        P(mutex)
        plato = platos[ocupado]
        ocupado = (ocupado+1) mod P
        V(mutex)
        V(vacio)
        repartir(plato)
    }
}
```

Passing the baton

Siempre que se nos diga que un recurso se debe de usar de manera ordenada

Sin coordinador

Hay que preguntar si la cola es vacía y si lo está no se encola

Enunciados de ejemplo:

- Ejercicio 6b de la práctica

Implemente una solución suponiendo que existe una única impresora compartida por todas las personas, y las mismas la deben usar de a una persona a la vez para el caso en que se deba respetar el orden de llegada.

- Ejercicio parcial 10-10-2023
  En una estación de trenes, asisten P personas que deben realizar una carga de su tarjeta SUBE en la terminal disponible. La terminal es utilizada en forma exclusiva por cada persona de acuerdo con el orden de llegada. Implemente una solución utilizando sólo procesos Persona

Ejercicio resuelto

(ejercicio 6b de la práctica)

```
74  Cola c
75  Sem waiting[P] = {[P]0}
76  Sem mutex = 1
77  bool vacio = true
78  Process Persona[id:1..N]{
79      P(mutex)
80      if (!vacio)
81          c.push(id)
82          V(mutex)
83          P(waiting[id])
84      else
85          vacio = false
86          V(mutex)
87      Imprimir(documento)
88      P(mutex)
89      if(c.empty())
90          vacio = true
91      else
92          V(waiting[c.pop()])
93      V(mutex)
94  }
```

Con coordinador

Hay que si o si encolarse

Enunciados de ejemplo:

- Ejercicio 6d de la práctica
  Modifique la solución del 6b para el caso en que además hay un proceso Coordinador que le indica a cada persona que es su turno de usar la impresora.

Ejercicio resuelto

(ejercicio 6d de la práctica)

```
Sem mutex = 1, libre = 0, contenido = 0
Sem waiting[N] = {[N]0}
Cola c
Process Persona[id:1..N]{
    P(mutex)
    c.push(id)
    V(contenido)
    V(mutex)
    P(waiting[id])
    imprimir(documento)
    V(libre)
}
Process Coordinador{
    for (i=0, i<N, i++){
        P(contenido)
        P(mutex)
        V(waiting[c.pop()])
        V(mutex)
        P(libre)
    }
```

Con coordinador y colas múltiples

Hay que tener una vector contador para cada cola

Enunciados de ejemplo:

- Ejercicio 12a de la práctica
  Simular la atención en una Terminal de Micros que posee 3 puestos para hisopar a 150 pasajeros. En cada puesto hay una Enfermera que atiende a los pasajeros de acuerdo con el orden de llegada al mismo.

Cuando llega un pasajero se dirige al Recepcionista, quien le indica qué puesto es el que tiene menos gente esperando. Luego se dirige al puesto y espera a que la enfermera correspondiente lo llame para hisoparlo. Finalmente, se retira.
a) Implemente una solución considerando los procesos Pasajeros, Enfermera y Recepcionista.

- Ejercicio parcial 11-10-22
  En una planta verificadora de vehículos, existen 7 estaciones donde se dirigen 150 vehículos para ser verificados. Cuando un vehículo llega a la planta, el coordinador de la planta le indica a qué estación debe dirigirse. El coordinador selecciona la estación que tenga menos vehículos asignados en ese momento. Una vez que el vehículo sabe qué estación le fue asignada, se dirige a la misma y espera a que lo llamen para verificar. Luego de la revisión, la estación le entrega un comprobante que indica si pasó la revisión o no. Más allá del resultado, el vehículo se retira de la planta.

Ejercicio resuelto

(ejercicio 12a de la práctica)

```
Sem mutexcola = 1, despertarRecepcionista = 0, hisopados[150] = {[150]0}, mutexenfermera[3] = {[3]1}, mutexvector = 1
Cola cola
Cola enfermeras[3]
int vector[3] = {[3]0}
Process pasajero[id:1..150]{
    P(mutexcola)
    cola.push(id)
    V(mutexcola)
    V(despertarRecepcionista)
    P(hisopados[id])
}
Process recepcionista{
    for(i=0,i<150,i++){
        P(despertarRecepcionista)
        P(mutexcola)
        id = cola.pop()
        V(mutexcola)

        P(mutexvector)
        idEnfermera = min(vector)
        vector[idenfermera]++
        V(mutexvector)

        P(mutexenfermera[idenfermera])
        enfermeras[idEnfermera].push(id)
        V(mutexenfermera[idenfermera])

        V(despertarenfermera[idenfermera])
    }
}
```

```
Process enfermera[id:1..3]{
    while (true){
        P(despertarenfermera[id])
        P(mutexenfermera[id])
        id = enfermeras[id].pop()
        V(mutexenfermera[id])
        id.Hisopar()
        V(hisopados[id])

        P(mutexvector)
        vector[id]--
        V(mutexvector)
    }
}
```

Sin coordinador y N recursos

Se debe hacer el passing the baton de la persona y del recurso

Enunciados de ejemplo:

- Ejercicio b del parcial 10-10-2023
  En una estación de trenes, asisten P personas que deben realizar una carga de su tarjeta SUBE en la terminal disponible. La terminal es

utilizada en forma exclusiva por cada persona de acuerdo con el orden de llegada. Considerando que hay T terminales disponibles. Las personas realizan una única fila y la carga la realizan en la primera terminal que se libera. Recuerde que sólo debe emplear procesos Persona. Nota: la función UsarTerminal(t) le permite cargar la SUBE en la terminal t.

Ejercicio resuelto

Ejercicio b del parcial 10-10-2023

```
Sem mutex1 = 1
Sem waiting[P] = {[P]0}
Cola cola
Cola terminales
int T[P]
int libres = T
Process Persona[id: 1..P]{
    p(mutex)
    if(libres = 0)
        cola.push(id)
        v(mutex)
        p(waiting[id])
    else
        libres--
        t[id] = terminales.pop()
        v(mutex)
    UsarTerminal(t)
    p(mutex)
    if(cola.empty())
        libres++
        terminales.push(t)
    else
        sig = cola.pop()
        t[sig] = t[id]
        v(waiting[sig])
    v(mutex)
```

Barrera

Cuando se debe esperar a que N procesos lleguen a un punto para seguir con la ejecución

Enunciados de ejemplo:

- Ejercicio 7 de la práctica
  Suponga que se tiene un curso con 50 alumnos. Cada alumno debe realizar una tarea y existen 10 enunciados posibles. Una vez que todos los alumnos eligieron su tarea, comienzan a realizarla. Cada vez que un alumno termina su tarea, le avisa al profesor y se queda esperando el puntaje del grupo (depende de todos aquellos que comparten el mismo enunciado). Cuando un grupo terminó, el profesor les otorga un puntaje que representa el orden en que se terminó esa tarea de las 10 posibles.
  Nota: Para elegir la tarea suponga que existe una función elegir que le asigna una tarea a un alumno (esta función asignará 10 tareas diferentes entre 50 alumnos, es decir, que 5 alumnos tendrán la tarea 1, otros 5 la tarea 2 y así sucesivamente para las 10 tareas).

- Ejercicio 11 de la práctica
  En un vacunatorio hay un empleado de salud para vacunar a 50 personas. El empleado de salud atiende a las personas de acuerdo con el orden de llegada y de a 5 personas a la vez. Es decir, que cuando está libre debe esperar a que haya al menos 5 personas esperando, luego vacuna a las 5 primeras personas, y al terminar las deja ir para esperar por otras 5. Cuando ha atendido a las 50 personas el empleado de salud se retira. Nota: todos los procesos deben terminar su ejecución

- Ejercicio parcial 13-12-22
  En una fábrica de muebles trabajan 50 empleados. Al llegar, los empleados forman 10 grupos de 5 personas cada uno, de acuerdo al orden de llegada (los 5 primeros en llegar forman el primer grupo, los 5 siguientes el segundo grupo, y así sucesivamente). Cuando un grupo se ha terminado de formar, todos sus integrantes se ponen a trabajar. Cada grupo debe armar M muebles (cada mueble es armado por un solo empleado); mientras haya muebles por armar en el grupo los empleados los irán resolviendo (cada mueble es armado por un solo empleado). Nota: Cada empleado puede tardar distinto tiempo en armar un mueble. Sólo se pueden usar los procesos “Empleado”, y todos deben terminar su ejecución.

Ejercicio resuelto

Ejercicio 7 de la práctica

```
Cola c
int cant = 0
Sem mutex1 = 1, mutex2 = 1, hayTrabajo = 0, barreraAlumnos = 0
Sem puntaje[10]
List nota[10]
Process Alumno[id:1..50]{
    int tarea, nota
    tarea = elegir()
    P(Mutex)
    cant++
    if(cant == 50){
        for (i=0, i<50, i++){
            V(barreraAlumnos)
        }
    }
    V(mutex)
    P(barreraAlumnos)
    //hacer tarea
    P(mutex2)
    c.push(tarea)
    V(mutex2)
    V(hayTrabajo) //despierto profesor
    P(puntaje[tarea])
    nota = nota[tarea]
}
```

```
Process Profesor{
    int nota = 10
    int tarea
    list grupos[10] = {[10]0}
    for (i=0, i<50, i++)
        P(hayTrabajo)
        P(mutex2)
        tarea = c.pop()
        V(mutex2)
        grupos[tarea]++ //cant de alumnos que terminaron x cada tarea
        if(grupos[tarea] == 5) //si llegó a 5 se les da la nota
            nota[tarea] = nota
            for (i=0, i<50, i++){
                V(puntaje[tarea])//despierto a todos los que tenian la tarea
            }
            nota--
}
```

P personas tienen que realizar T cosas

El título se explica solo

Enunciados de ejemplo:

- Ejercicio 8 de la práctica
  Una fábrica de piezas metálicas debe producir T piezas por día. Para eso, cuenta con E empleados que se ocupan de producir las piezas de a una por vez. La fábrica empieza a producir una vez que todos los empleados llegaron. Mientras haya piezas por fabricar, los empleados tomarán una y la realizarán. Cada empleado puede tardar distinto tiempo en fabricar una pieza. Al finalizar el día, se debe conocer cual es el empleado que más piezas fabricó.

- Ejercicio parcial 4-12-2023
  Un sistema debe validar un conjunto de 10000 transacciones que se encuentran disponibles en una estructura de datos. Para ello, el sistema dispone de 7 workers, los cuales trabajan colaborativamente validando de a 1 transacción por vez cada uno. Cada validación puede tomar un tiempo diferente y para realizarla los workers disponen de la función Validar(t), la cual retorna como resultado un número entero entre 0 al 9. Al finalizar el procesamiento, el último worker en terminar debe informar la cantidad de transacciones por cada resultado de la función de validación. Nota: maximizar la concurrencia.

- Ejercicio parcial 13-12-2022
  En una fábrica de muebles trabajan 50 empleados. A llegar, los empleados forman 10 grupos de 5 personas cada uno, de acuerdo al orden de llegada (los 5 primeros en llegar forman el primer grupo, los 5 siguientes el segundo grupo, y así sucesivamente). Cuando un grupo se ha terminado de formar, todos sus integrantes se ponen a trabajar. Cada grupo debe armar M muebles (cada mueble es armado por un solo empleado); mientras haya muebles por armar en el grupo los empleados los irán resolviendo (cada mueble es armado por un solo empleado). Nota: Cada empleado puede tardar distinto tiempo en armar un mueble. Sólo se pueden usar los procesos “Empleado”, y todos deben terminar su ejecución. Maximizar la concurrencia.

Ejercicio resuelto

Ejercicio 8 de la práctica

```
int empleados = 0, finalizados = 0, e
Sem mutex1 = 1, mutex2 = 1, mutex3 = 1, barreraEmpleados = 0, despertarEmpresa = 0, listo = 0
list empleados[E] = {[E]0}
Process Empleado[id:1.E]{
    //llega
    P(mutex)
    empleados++
    if (empleados == E)
        for (i=0, i<E, i++){
            V(barreraEmpleados)
        }
    V(mutex)
    P(barreraEmpleados)
    P(mutex2)
    while(piezas < T){
        piezas++
        V(mutex2)
        //labura
        P(mutex2)
        empleados[id]++ //para saber cuanto trabajó cda empleado
    }
    V(mutex2)
    P(mutex3)
    finalizados++
    if (finalizados == E)
        V(despertarEmpresa)
    V(mutex3)
    P(listo)
    if(id == e)
        //soy el mas capo
    else
        //no soy el mas capo
}
```

```
Process Empresa{
    P(despertarEmpresa)
    e = empleados.mayor()
    for (i=0, i<E, i++){
        V(listo)
    }
}
```

Tipos de ejercicios de monitores

Passing the condition

Siempre que se deba usar un recurso de manera ordenada

Sin coordinador

Enunciados de ejemplo:

- Ejercicio 3b de la práctica
  Existen N personas que deben fotocopiar un documento. La fotocopiadora sólo puede ser usada por una persona a la vez y se debe respetar el orden de llegada

Ejercicio resuelto

```cpp
Monitor Fotocopiadora {
    bool libre = true;
    cond cola;
    int esperando = 0;
    Procedure usar (){
        if (not libre){
            esperando++
            wait(cola)
        }
        else libre = false
    }
    Procedure salir(){
        if (esperando > 0){
            esperando--
            signal(cola)
        }
        else libre = true
    }
}

Process Persona [id:1..N]{
    int edad
    Fotocopiadora.usar(id, edad);
    //fotocopiar
    Fotocopiadora.salir();
```

Con coordinador

Los procesos se deben esperar mutuamente

Enunciados de ejemplo:

- Ejercicio 3e de la práctica
  Modifique la solución de (b) para el caso en que además haya un Empleado que le indica a cada persona cuando debe usar la fotocopiadora.

Ejercicio resuelto

```
148  Monitor Fotocopiadora {
149      cond persona;
150      cond empleado;
151      cond termine
152      int esperando = 0;
153      Procedure asignar(){
154          if (esperando == 0)
155              wait(empleado)
156          esperando--
157          signal(persona)
158          wait(termine)
159      }
160      Procedure usar (){
161          signal(empleado)
162          esperando++
163          wait(persona)
164      }
165      Procedure salir(){
166          signal(termine)
167      }
168  }
169
170  Process Persona [id:1..N]{
171      fotocopiadora.usar()
172      //fotocopiar
173      fotocopiadora.dejar()
174  }
175
176   Process Empleado {
177      for (i=1; i<=N; i++)
178          fotocopiadora.asignar()
179  }
```

Grupos de prioridades

No está bueno asumir que una cola ordena por grupo de prioridad y orden de llegada, entonces o bien puedo hacer colas para cada prioridad o llevar

un contador dependiendo del ejercicio. Ya no se despierta por nro de id sino por prioridad.
Esto solo aplica a GRUPOS de prioridad. Por ejemplo en el ejercicio 3c de la práctica te pide que se ordene por edad, en ese caso si se usa una cola ordenada.

Enunciados de ejemplo:

- Ejercicio parcial 10-10-2023
  En una elección estudiantil, se utiliza una máquina para voto electrónico. Existen N Personas que votan y una Autoridad de Mesa que les da acceso a la máquina de acuerdo con el orden de llegada, aunque ancianos y embarazadas tienen prioridad sobre el resto. La máquina de voto sólo puede ser usada por una persona a la vez

Ejercicio resuelto

Ejercicio parcial 10-10-2023

```
Monitor Máquina{
    cond autoridad, maquinaLibre
    cond esperaP, esperaS
    int cantP == 0, cantS == 0
    Procedure llegar(id, prioridad: in bool){
        if(prioridad){
            cantP++
            signal(autoridad)
            wait(esperaP)
        }
        else{
            cantS++
            signal(autoridad)
            wait(esperaS)
        }
    }
    Procedure darAcceso(){
        if(cantP == 0 && cantS == 0)
            wait(autoridad)
        elif(cantP > 0)
            cantP--
            signal(esperaP)
        else
            cantS--
            signal(esperaS)
        wait(maquinalibre)
    }
    Procedure dejar(){
        signal(maquinaLibre)
    }
}
```

```
Process Persona[id:1..N]{
    bool prioridad
    maquina.llegar(id, prioridad)
    votar()
    maquina.dejar()
}
Process Autoridad{
    for(i=0, i<N, i++){
        maquina.darAcceso()
    }
}
```

Monitor de acceso y monitor de uso sincronizado

Para maximizar la concurrencia a veces es preferible hacer 2 monitores: uno que controle el acceso (generalmente a la cola) y otro para realizar una acción sincronizada entre procesos. De esta manera mientras se realiza el uso sincronizado se puede encolar al mismo tiempo.

Enunciados de ejemplo

- Ejercicio parcial recupertatorio
  Resolver con MONITORES el siguiente problema. En una planta verificadora de vehículos existen 5 estaciones de verificación. Hay 75 vehículos que van para ser verificados, cada uno conoce el número de estación a la cual debe ir. Cada vehículo se dirige a la estación correspondiente y espera a que lo atiendan. Una vez que le entregan el comprobante de verificación, el vehículo se retira. Considere que en cada estación se atienden a los vehículos de acuerdo con el orden de llegada. Nota: maximizar la concurrencia.
- Ejercicio 7 de la práctica
  Se debe simular una maratón con C corredores donde en la llegada hay UNA máquina expendedoras de agua con capacidad para 20 botellas. Además, existe un repositor encargado de reponer las botellas de la máquina. Cuando los C corredores han llegado al inicio comienza la carrera. Cuando un corredor termina la carrera se dirigen a la máquina expendedora, espera su turno (respetando el orden de llegada), saca una botella y se retira. Si encuentra la máquina sin botellas, le avisa al repositor para que cargue nuevamente la máquina con 20 botellas; espera a que se haga la recarga; saca una botella y se retira. Nota: mientras se reponen las botellas se debe permitir que otros corredores se encolen.

Ejercicio resuelto

Ejercicio 7 de la práctica

```
643     process Corredor[id:=1..C]{
644         Carrera.Iniciar();
645         //Corre carrera
646         Carrera.AccesoMaquina();
647         Maquina.TomarBotellita();
648         CarrerA.SiguienteMaquina();
649     }
```

```
587     Monitor Carrera {
588         int esperando;
589         cond salir;
590         procedure Iniciar(){
591             esperando++;
592             if esperando == C {
593                 signal_all(salir);
594             } else {
595                 wait(salir);
596             }
597         }
598
599         procedure AccesoMaquina(){
600             if(maquinaLibre){
601                 maquinaLibre = false;
602             } else {
603                 esperando++;
604                 wait(despertarEsperando);
605             }
606         }
607
608         procedure SiguienteMaquina(){
609             if(esperando > 0){
610                 esperando--;
611                 signal(despertarEsperando)
612             } else {
613                 maquinaLibre = true;
614             }
```

```
617
618     Monitor Maquina{
619         int cantBotellas = 20; cond hayBotellas;
620         bool requieroReposicion = false;
621         procedure Reponer(){
622             if(!requieroReposicion){
623                 wait(despertarRepositor)
624             }
625             cantBotellas = 20;
626             signal(hayBotellas);
627         }
628
629         procedure TomarBotellita(in int id){
630             if(cantBotellas == 0){
631                 requieroReposicion = true;
632                 signal(despertarRepositor);
633                 wait(hayBotellas);
634             }
635             cantBotellas--;
636         }
637     }
638     process Repositor{
639         while(true){
640             Maquina.Reponer();
641         }
642     }
```

Barrera

Barrera para un mismo tipo de proceso

El último que llega despierta a todos

Enunciados de ejemplo:

- Ejercicio Parcial 4-12-2023
  En una empresa trabajan 20 vendedores ambulantes que forman 5 equipos de 4 personas cada uno (cada vendedor conoce previamente a que equipo pertenece). Cada equipo se encarga de vender un producto diferente. Las personas de un equipo se deben juntar antes de comenzar a trabajar. Luego cada integrante del equipo trabaja independientemente del resto vendiendo ejemplares del producto correspondiente. Al terminar cada integrante del grupo debe conocer la cantidad de ejemplares vendidos por el grupo. Nota: maximizar la concurrencia.

Ejercicio resuelto

Ejercicio de parcial 4-12-2023

```
43  Process Vendedores[id:1..20]{
44      equipo[id].reunirse()
45      cantVendidos = vender()
46      equipo[id].terminar(cantVendidos, total)
47  }
48  Monitor Equipos[id:1..5]{
49      int esperando = 0, terminados = 0, result = 0
50      cond grupo
51      Procedure reunirse(){
52          esperando++
53          if(esperando < 4){
54              wait(grupo)
55          }
56          else{
57              signal_all(grupo)
58          }
59      }
60      Procedure terminar(cantVendidos: in int, total: out int){
61          terminados++
62          result += cantVendidos
63          if(terminados < 4){
64              wait(grupo)
65          }
66          else{
67              signal_all(grupo)
68          }
69          total = result
70      }
71  }
```

Barrera para despertar procedure llamado por otro proceso

El proceso que libera la barrera para que el otro proceso pueda ejecutar el monitor también debe de dormirse

Enunciados de ejemplo:

- Ejercicio 6 de la práctica
  Existe una comisión de 50 alumnos que deben realizar tareas de a pares, las cuales son corregidas por un JTP. Cuando los alumnos llegan, forman una fila. Una vez que están todos en fila, el JTP les asigna un número de grupo a cada uno. Para ello, suponga que existe una función AsignarNroGrupo() que retorna un número “aleatorio” del 1 al 25. Cuando un alumno ha recibido su número de grupo, comienza a realizar su tarea. Al terminarla, el alumno le avisa al JTP y espera por su nota. Cuando los dos alumnos del grupo completaron la tarea, el JTP les asigna un puntaje (el primer grupo en

terminar tendrá como nota 25, el segundo 24, y así sucesivamente hasta el último que tendrá nota 1)

- Ejercicio 8 de la práctica
  En un entrenamiento de fútbol 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).
- Ejercicio 9 de la práctica
  En un examen de la secundaria hay un preceptor y una profesora que deben tomar un examen escrito a 45 alumnos. El preceptor se encarga de darle el enunciado del examen a los alumnos cundo los 45 han llegado (es el mismo enunciado para todos). La profesora se encarga de ir corrigiendo los exámenes de acuerdo con el orden en que los alumnos van entregando. Cada alumno al llegar espera a que le den el enunciado, resuelve el examen, y al terminar lo deja para que la profesora lo corrija y le envíe la nota.

Ejercicio resuelto

Ejercicio 8 de la práctica

```
Process Jugador[id:1..20]{
    equipo[DarEquipo()].esperar(c)
    cancha[c].esperar()
}

Process Partido[id:1..2]{
    cancha[id].jugar()
    dalay(50)
    cancha[id].terminar()
}

Monitor Equipo[id:1..4]{
    int cantJugadores = 0
    cond jugadores, listo
    Procedure llegar(cancha: out int){
        cantJugadores++
        if(cantJugadores == 5)
            asignar.equipoListo(cancha)
            signal_all(jugadores)
        else
            wait(jugadores)
    }
}
```

```
Monitor Asignar{
    int num = 1
    int esperando = 0
    cond otroEquipo
    Procedure equipoListo(cancha: out int){
        if(esperando == 1)
            signal(otroEquipo)
        else
            esperando++
        if (esperando <= 1)
            wait(otroEquipo)
        esperando--
        cancha = num
        num++
    }
}

Monitor Cancha[id:1..2]{
    int cantJugadores = 0
    Procedure esperar(){
        canJugadores++
        if (cantJugadores == 10)
            signal(inicio)
        wait(jugadores)
    }
    Procedure jugar(){
        if(cantJugadores < 10)
            wait(inicio)
    }
    Procedure terminar(){
        signal_all(jugadores)
    }
}
```