Resolución de practica de monitores

Ejercicio 1

Se dispone de un puente por el cual puede pasar un solo auto a la vez. Un auto pide permiso para pasar por el puente, cruza por el mismo y luego sigue su camino.

```
Monitor  Puente
    cond cola;
    int cant= 0;

    Procedure entrarPuente ()
        while ( cant > 0) wait (cola);
        cant = cant + 1;
    end;

    Procedure salirPuente ()
        cant = cant – 1;
        signal(cola);
    end;
End Monitor;

Process Auto [a:1..M]
    Puente. entrarPuente (a);
    "el auto cruza el puente"
    Puente. salirPuente(a);
End Process;
```

a. ¿El código funciona correctamente? Justifique su respuesta.
b. ¿Se podría simplificar el programa? ¿Sin monitor? ¿Menos procedimientos? ¿Sin variable condition? En caso afirmativo, rescriba el código.

c. ¿La solución original respeta el orden de llegada de los vehículos? Si rescribió el código en el punto b), ¿esa solución respeta el orden de llegada?

Respuesta :

a. Es correcta, solo lo del parámetro.
b. El monitor representa el recurso compartido que es el puente.
Lo único que se debe asegurar es la exclusión mutua la cual los monitores ya la manejan por definición, ya que no va a entrar más procedimiento a la vez.
La exclusión mutua es implícita dentro de un monitor al no poder ejecutar mas de un llamado a un procedimiento a la vez, hasta que no se termina el procedure o no se duerme en una variable condition no se libera el monitor para atender otro llamado.

```
Monitor puente{

    procedure entrarPuente(int a){
        usarPuente()
    }
}

process Auto[id:0..N]{
    Puente.entrarPuente(id)
}
```

c. La primera solución no respeta el orden de llegada, ya que cuando se despierten los procesos van a competir por el monitor.
La segunda no respeta el orden de llegada, ya que no sabemos quienes son los que llegan primero ya que competirán por el recurso (Para que se respete el orden se debe usar Passing the condition).

Ejercicio 2

Existen N procesos que deben leer información de una base de datos, la cual es administrada por un motor que admite una cantidad limitada de consultas simultáneas.

a) Analice el problema y defina qué procesos, recursos y monitores/sincronizaciones serán necesarios/convenientes para resolverlo.
b) Implemente el acceso a la base por parte de los procesos, sabiendo que el motor de base de datos puede atender a lo sumo 5 consultas de lectura simultáneas.

Respuesta :

a. Se necesitara un monitor que represente el recurso que es la base de datos, un proceso que se encargue de administrar el acceso. También el procedimiento que solicite el acceso..... COMPLETAR.

b.

```
Monitor BD{
    cond espera;
    int cantidad = 0;

    procedure entrar(){
        while(cant == 5)  wait(espera)
            // permitir el acceso
            cantidad ++
    }

    procedure salir(){
        cantidad --
        signal(espera)
    }

}

process lectores[id: 0..N-1]{
    BD.entrar()
    // Hace sus cosas
    BD.salir()
}
```

Ejercicio 3 - Consultar a,d,e,f

Existen N personas que deben fotocopiar un documento. La fotocopiadora sólo puede ser usada por una persona a la vez. Analice el problema y defina qué procesos, recursos y monitores serán necesarios/convenientes, además de las posibles sincronizaciones requeridas para resolver el problema. Luego, resuelva considerando las siguientes situaciones:

a) Implemente una solución suponiendo no importa el orden de uso. Existe una función Fotocopiar() que simula el uso de la fotocopiadora.
b) Modifique la solución de (a) para el caso en que se deba respetar el orden de llegada.
c) Modifique la solución de (b) para el caso en que se deba dar prioridad de acuerdo con la edad de cada persona (cuando la fotocopiadora está libre la debe usar la persona de mayor edad entre las que estén esperando para usarla).

d) Modifique la solución de (a) para el caso en que se deba respetar estrictamente el orden dado por el identificador del proceso (la persona X no puede usar la fotocopiadora hasta que no haya terminado de usarla la persona X-1).
e) 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.
f) Modificar la solución (e) para el caso en que sean 10 fotocopiadoras. El empleado le indica a la persona cuál fotocopiadora usar y cuándo hacerlo.

Respuesta :

```
// A.  Preguntar como hacemos el pasaje de parametros y si es necesario

Monitor fotocopiadora{
    procedure usar(){
        Fotocopiar()
    }

}

process persona[id: 0..N-1]{
    fotocopiadora.usar()
}
```

```
// B
// Para que se respete el orden debemos usar passing the condition

Monitor fotocopiadora{

    bool lire = true
    cond cola;
    int esperando = 0

    Procedure Usar(){
        if (not libre) {
            esperando ++
            wait(cola)
        }
    }

    procedure Salir(){
        if ( esperando > 0 ){
```

```
            esperando --
            signal(cola)
        }
    }

}

process persona[id:0..N-1]{
    fotocopiadora.Usar()

    Fotocopiar()        // Acá esta usando la función Fotocopiar()

    fotocopiadora.Salir()
}
```

```
// C

Monitor fotocopiadora{

    bool libre = true
    colaOrdenada fila;
    int esperando = 0
    cond espera[N];


    Procedure Usar(int in edad, in id){
        if (not libre) {
            fila.push(id, edad)
            esperando ++
            wait(espera[id])
        }
    }

    procedure Salir(){
        if ( esperando > 0 ){
            esperando --
            fila.pop(id)
            signal(espera[id])
        }
    }

}
```

```
process persona[id:0..N-1]{
    fotocopiadora.Usar(edad,id)

    Fotocopiar()        // Acá esta usando la función Fotocopiar()

    fotocopiadora.Salir()
}
```

```
// D
// Se debe usar un arreglo de cond para poder dormir a los procesos

Monitor fotocopiadora{


    int sig = 0
    cond espera[n];

    procedure usar(int in id){
        if (id != sig )  wait(espera[id]) // Acá quiero preguntar xq no es un
while
            //
    }

    procedure salir(){
        sig ++
        signal(espera[sig])
    }

}

process persona[id: 0..N-1]{
    fotocopiadora.usar(id)

    Fotocopiar()

    fotocopiadora().salir()
}
```

```
// E

// Para que se respete el orden debemos usar passing the condition
```

```
// Pasa a ser como el  ejercico 6 , passing the condition
// Parece el ejercicio 6, podemos verlo .....


// En este caso no es necesario una cola.Ya que se duermen en una variable de
condición.

// Cuando el orden es diferente el orden de llega si necesitas una cola

Monitor fotocopiadora{

    bool libre = true
    int espera = 0

    Cola c;
    cond espera;
    int cantidad;

    cond domirse;
    cond HayPersona;

    Procedure PedidoUso(int id){
        cantidad++;
        wait(esperaUso)
    }

    procedure Sig(){
        if (not libre){
            wait(dormirse) // Se duerme el empleado hasta que el recurso este
libre
        }else{
            if (espera = 0) {
                wait(HayPersona) // Espero a que haya alguien
            }

        signal(esperaUso)
        libre = false // Al momento de que la usa, la ocupa y
        cantidad-- // Despues del signal se hace
    }

    procedure salir(){
        libre = true  // Dejo la variable en libre
        signal(dormirse) // Le avisa al empleado que ya salio
    }
```

```
}

process persona[id:0..N-1]{
    fotocopiadora.PedidoUso(id)
    Fotocopiar()        // Acá esta usando la función Fotocopiar()
    fotocopiadora.Salir()
}

process empleado{
    while (true){ // Podria ser un for pero decis que solo hace una fotocopia.
        fotocopiadora.sig()
    }

}

// Si fuesen dos deberia usar el while
```

> Si la persona tiene que usar cualquier impresora, no puedo tener 10 impresoras ya que no sabria donde ir . En caso de que cada impresora tiene su empleado puede ser que si

> Cuando la persona no tiene que pedir una especifica hay un solo empleado que se encarga de eso.
> Para que sea más de un monitor , necesitan saber a cual ir, como el de las canchas, ya sabe a cual cancha ir. En este caso no sabe a que impresora ir .

```
//F
```

Ejercicio 4 - Preguntar por la lógica de como hacerlo

Existen N vehículos que deben pasar por un puente de acuerdo con el orden de llegada. Considere que el puente no soporta más de 50000kg y que cada vehículo cuenta con su propio peso (ningún vehículo supera el peso soportado por el puente).

```
// Acá el tema es el orden de llegada

Monitor puente{

    bool libre = true;
    int pesoActual = 0
    colaOrdenada fila;
```

```
    cond esperando;
    cond sali;

    procedure pasar(int in peso, int in id){
        // Acá si no cumple el if debe dormirse en otra variable extra ya que
si se vuelve a dormir se va al final
        if ( (pesoActual + peso) > 50.000)
            wait(espera)

// ------------------------------------------
    //Bien




    cola.push(id,peso); // Llego y me encolo
    signal(llegoAuto)  // Aviso que llegue
    wait(esperando);   //Esperamos a que nos dejen pasar
    pesoActual+= peso // Cuando me deja pasar incremento mi peso
    }









    procedure salir(int in peso){
        // MAAAAL
        pesoActual -= peso
        if ( pesoActual == 0 ) // Soy el ultimo
            //Acá hace
        else

        //signal(espera)
        // ve y si cumple el peso le hace el signal

// -------------------------
    // Bien
```

```
    pesoaActual =- peso;
    while (pesoActual < 50.000){
        verPeso(cola,peso) // Ve el peso del proximo
        if ( (peso + pesoActual ) < 50.000){  // Si puede pasar
            signal(esperando)  // Lo despierta
            cola.pop() // Lo saca el auto que cumple
        }else{
            wait(sali) // Se duerme hasta que alguien le avise que salio
        }
    }
}

process Vehiculo[id:0..N-1]{
int peso;
    Puente.pasar(peso)
    // Cruza el puente
    Puente.salir(peso)


}
```

```
Monitor puente{

    int pesoActual = 0
    c cola;

    int cantAutos = 0;
    cond esperando;
    cond sali;

    procedure pasar(int in peso){
        if ((peso + pesoActual ) < 50.000) and isEmpty(cola) ) {
            pesoaActual += peso;
        }else{
            // Si no puedo entrar me duermo
            cola.push(peso); // Llego y me encolo
```

```
            //signal(llegoAuto)  // Aviso que llegue
            wait(esperando);   //Esperamos a que nos dejen pasar
            //pesoActual+= peso // Cuando me deja pasar incremento mi peso
        }

    }

    procedure salir(int in peso){
    pesoaActual -= peso;
    seguir = true
    while (seguir){
        // Si no esta vacia......
        verPeso(cola,peso) // Ve el peso del proximo
        if ( (peso + pesoActual ) < 50.000){  // Si puede pasar
            peso = cola.pop() // Saca el auto que cumple
            pesoActual += peso
            signal(esperando)  // Lo despierta
        }else{
            seguir = false
            //wait(sali) // Se duerme hasta que alguien le avise que salio
        }
    }
}

process Vehiculo[id:0..N-1]{
int peso;
    Puente.pasar(peso)
    // Cruza el puente
    Puente.salir(peso)


}
```

Ejercicio 5

En un corralón de materiales se deben atender a N clientes de acuerdo con el orden de llegada. Cuando un cliente es llamado para ser atendido, entrega una lista con los productos que comprará, y espera a que alguno de los empleados le entregue el comprobante de la compra realizada.

a) Resuelva considerando que el corralón tiene un único empleado.
b) Resuelva considerando que el corralón tiene E empleados (E > 1). Los empleados no deben terminar su ejecución.
c) Modifique la solución (b) considerando que los empleados deben terminar su ejecución cuando se hayan atendido todos los clientes.

Respuesta :

```
// A


Monitor Gestion{

    cond espera;
    cond espera_trabajo;

    cond hayPersona;
    int vector([N])

    cola c;

    procedure llegue(int in id, lista in l, int outcomprobante ){
        c.push(id,l)
        signal(HayPersona)
        wait(espera) //Dormir el cliente hasta que le den el resultado
        comprobante = vector[id]
    }

    procedure sig(int out id, lista out l){   // Lo hace el emleado
        if (c.isEmpty()){
            wait(hayPersona)
        c.pop(id,l)//Desencola al sig
    }

    procedure atender(int in id, lista out l){
        v[id] = comprobante
        signal(espera) //Le aviso que ya esta su comprobante
    }
}

process cliente(id: 0..N-1){
    Gestion.llegue(id,l,comprobante)
}

process Empleado{
int id; lista l;
int comprobante;
    for i: 1 to N
        Gestion.sig(id,l)   //Espera a que haya el siguiente
        comprobante = generarComprobante(l)
        Gestion.atender(id,comprobante)
```

```
}
```

```
//B
Monitor Gestion{

    cond espera;
    cond espera_trabajo;

    cond hayPersona;
    int vector([N])

    cola c;

    procedure llegue(int in id, lista in l, int outcomprobante ){
        c.push(id,l)
        signal(HayPersona)
        wait(espera) //Dormir el cliente hasta que le den el resultado
        comprobante = vector[id]
    }

    procedure sig(int out id, lista out l){   // Lo hace el emleado
        if (c.isEmpty()){
            wait(hayPersona)
        c.pop(id,l)//Desencola al sig
    }

    procedure atender(int in id, lista out l){
        v[id] = comprobante
        signal(espera) //Le aviso que ya esta su comprobante
    }
}

process cliente(id: 0..N-1){
    Gestion.llegue(id,l,comprobante)
}

process Empleado[id: 0..N-1]{
int id; lista l;
int comprobante;
    while (True)
        Gestion.sig(id,l)   //Espera a que haya el siguiente
        comprobante = generarComprobante(l)
        Gestion.atender(id,comprobante)
```

```
}
```

Ejercicio 6 (Pasar logica del monitor a los procesos) (jtp monitor)un solo monitor mejor y dos procesos

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). Nota: el JTP no guarda el número de grupo que le asigna a cada alumno.

```
Monitor Administrador{
    cola c;
    int tot = 0;
    int alumnos[50];
    cond esperarNumeroGrupo[50];
    cond somosCincuenta;

    cola terminoId;
    cond termine;
    cond terminoGrupo[25];          //Para saber que ya esta nuestra nota
    int grupoCantTermino([25] 0) ; //Para contar si terminaron los dos
    int grupoNota[25];             //Para la nota
    int nota = 25;

    //-----------------------------------------------

    process llegue(int in id, int out grupo){
        tot++;        //Incrementa la cantidad total que hay
        c.push(id);   //Se encola
        if ( tot == 50){
                signal(somosCincuenta); //Le avisa que ya llegaron todos
        wait(esperarNumeroGrupo[id])   //Espera poder arrancar
        grupo = alumnos[id]            //Se guarda el grupo que le toco
    }


    process esperarLlegada( ){
        if (tot < 50)
```

```
            wait(somosCincuenta);
        for i : 1 to 25
            numeroGrupo = AsignarNumeroGrupo();
            for i: 1 to 2
                alumnoId = c.pop()  //Saco el id del alumno (ya que es como
llegan)
                alumnos[alumnoId] = numeroGrupo; //Le asigna un grupo
                signal(esperarNumeroGrupo[alumnoId]);    //Lo despierta
    }


    // ------------------------------------------------------

    process termine(int in grupo, int out puntaje){
        terminoId.push(grupo);  //Me encolo en los que terminaron
        signal(Termine)         //Le aviso que termine
        wait(terminoGrupo[grupo])   //Espero a que termine mi grupo
        puntaje = grupoNota[grupo]  //Me guardo mi nota
    }

    procedure esperarQueTerminen(){
    if ( terminoId.isEmpty())
        wait(Termine)             //Espera a que le avisen que termino
    grupo = terminoId.pop()   //Saca el num del grupo del que termino
    grupoCantTermino[grupo] ++ //Incrementa
    if (grupoCantTermino[grupo] == 2){
        grupoNota[grupo] = nota;     //Les pone su nota
        signal_all(terminoGrupo[grupo])  // le avisa con el signal al grupo
        nota --;     //Decrementa la nota del proximo
            }
    }
}



process alumno[id: 1..50]{
int grupo, puntaje, id;
    Administrador.llegue(id,grupo);
    //Realiza la tarea
    Administrador.termine(grupo, puntaje);
}

process jtp{
int numeroGrupo
    Administrador.esperarLlegada();
    for i : 1 to 50
```

```
        Escritorio.esperarQueTerminen()
}
```

Ejercicio 7 - Consultar Bien la segunda

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.

Solución :

> Dudas :
>
> 1. La carrera es un monitor barrera ? pero cuando termina ?
> 2. La cola tiene que ir en el recurso de la expendedora ?
> 3. Acá dice que el corredor le avisa si no hay más botellas, cuando se va un corredor. Debe chequearlo ?

```
Monitor Expendedora{

    int cantBotellas = 20;
    int cantEsperando = 0;
    boolean libre = true;
    cond corredoresEsperando;
    cond quieroAgua;

    bool reponiendo = false;
    cond noHayAgua;

    procedure esperarUsoExpendedora(){
        if ((cantEsperando > 0) && libre ){
            wait(corredoresEsperando);  // Espero a poder pasar
        }else{
            libre = false; //Sino ya podes pasar y la marcas que no esta libre
        }

    }
```

```
    procedure retirarAgua(){
        if ( (cantBotellas = 0) ){
            signal(noHayAgua);      //Si no hay agua le avisa al repositor
            wait(yaHayAgua);        //Espera a que termine de cargar
        }
        cantBotellas --;                    //Retira el agua
        cantEsperando --;
        libre = true;                       //Pone que esta libre la maquina
        signal(corredoresEsperando);        //Es el turno del proximo
    }

    procedure esperandoRecargar(){
        wait(noHayAgua);        //Espera a que le avisen que no hay agua
        cantBotellas = 20;      //Lo carga
        signal(yaHayAgua);      //Avisa que ya se cargaron las 20 aguas
    }

}

process Corredores[id : 1.. C]{
    // Realiza la carrera
    //Termina la carrera
    Expendedora.esperarUsoExpendedora();
    Expendedora.retirarAgua();
}

process Repositor{
    while (True){
        Expendedora.esperandoRecargar();
    }
}
```

> Solución en el caso que se retire de la maratón recién cuando saca un agua.
> ¿En esperando recarga debe ir un if botellas > 0 wait ?

```
Monitor Carrera{
    int cantInicio = 0;
    cond esperandoCarrera;

    int cantEsperandoMaquina = 0;
    boolean libre = True;
```

```
    cond corredoresEsperando;

    procedure esperarInicio(){
        cant ++;
        if (cant == C){
            signal_all(esperandoCarrera); // Si soy el ultimo despierto a
todos
        }else{
            wait(esperandoCarrera);       //Me encolo y espero a que lleguen
todos
        }
    }

    procedure esperarUsoExpendedora(){
        //Verifico el cero ya que puede llegar a pasar alguien despues de mi
al ser distintos monitores.
        if ((cantEsperando == 0) && libre ){
            libre = false; //Podes pasar y la marcas que no esta libre
        }else{
            cantEsperando++;
            wait(corredoresEsperando);  // Espero a poder pasar
            //cantEsperando--; // cambiar
        }
    }

    procedure retirarse(){
        if (cantEsperando >  0 )
            cantEsperando--;
            signal(corredoresEsperando); //Que pase el sig
        } else {
            libre = true; // cambiar
        }
    }
}

Monitor Expendedora{
    int cantBotellas = 20;
    cond noHayAgua;
    cond yaHayAgua;

    procedure retirarAgua(){
        if ( (cantBotellas = 0) ){
            signal(noHayAgua);      //Si no hay agua le avisa al repositor
            wait(yaHayAgua);        //Espera a que termine de cargar
        }
        cantBotellas --;                    //Retira el agua
```

```
    }
    procedure esperandoRecargar(){
        if (cantBotellas > 0 )  // puede pasar que nunca se despierte ya que
primero pasan los 20 ----------
            wait(noHayAgua);       //Espera a que le avisen que no hay agua
        cantBotellas = 20;     //Lo carga
        signal(yaHayAgua);     //Avisa que ya se cargaron las 20 aguas
    }
}

process Corredores[id : 1.. C]{
    Carrera.esperarInicio();
    // Corre
    Carrera.esperarUsoExpendedora();
    Expendedora.retirarAgua();
    Carrera.retirarse();
}

process Repositor{
    while (True){
        Expendedora.esperandoRecargar();
    }
}
```

Ejercicio 8

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 - chequear a lo ultimo (Lo pensé con dos monitores) Bien

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 cuando 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. Nota: maximizar la concurrencia; todos los

procesos deben terminar su ejecución; suponga que la profesora tiene una función corregirExamen que recibe un examen y devuelve un entero con la nota.

Solución :

> Lo pense con dos monitores,

```
Monitor AdministradorExamen{
    int alumnosLlegaron = 0;
    cond soyElUltimo;
    cond esperarComienzo;

    txt vectorExamenes[45];

    procedure llegue(){
        alumnosLlegaron ++;
        if (alumnosLlegaron == 45){
            signal(soyElUltimo);  //Si soy el ultimo le aviso al preceptor
        }
        wait(esperarComienzo)  //Me duermo hasta que nos despierte (incluso el
ultimo)
    }
    procedure retirarExamen(int in id, txt out examen){
        examen = vectorExamenes[id];    //Busco mi examen
    }

    procedure repartirExamenes(txt in examen){
        if (alumnosLlegaron < 45 )
            wait(soyElUltimo); //Espera a que lleguen todos
        for i: 1 to 45 do
            vectoreExamenes[id] = examen; //Deja los examenes
        signal_all(esperarComienzo); //Despierta a todos los alumnos
    }

}


Monitor ColaCorreccion{
    cola c;
    int vectorNotas[45];
    cond esperaNota[45];
    cond entregoParcial;

    procedure entregarParcial(int in id,txt in examen, int out nota){
        c.push(id,examen);     //Encola su id y su examen
        signal(entregoPacial); //Aviso que entrego el parcial
```

```
        wait(esperaNota[id]);  //Espera a que este su nota
        nota = vectorNotas[id] //Guardo mi nota
    }
    }

    procedure esperarExamen(int out id,txt out examen){
        if (c.isEmpty()){
            wait(entregoParcial); //Espera hasta que alguien entregue
        //}else{ .. no va ya que cuando se despierta no lo saca
        c.pop(id,examen)      //Saca un examen
        }
    }

    procedure darNota(int in id,int in resultado){
        vectorNotas[id] = resultado; //Le deja el resultado en el vector
        signal(esperaNota[id]); //Le avisa que ya esta su nota
    }

}

process alumnos[id:1..45]{
txt examen;
int nota;
    AdministradorExamen.llegue();
    AdministradorExamen.retirarExamen(id, examen);
    //Realiza el examen
    ColaCorreccion.entregarParcial(id,examen,nota);
}

process profesora{
int id; txt examen; int resultado;
    for i : 1 to 45
        ColaCorreccion.esperarExamen(id, examen);
        resultado = CorregirExamen(examen);
        ColaCorreccion.darNota(id,resultado);
}

process preceptor{
txt examen;
    AdministradorExamen.repartirExamenes(examen);
}
```

Ejercicio 10 - Consultar - Esta mal, se traba en el empleado (Ya esta corregido, la ultima solución esta bien)

En un parque hay un juego para ser usada por N personas de a una a la vez y de acuerdo al orden en que llegan para solicitar su uso. Además, hay un empleado encargado de desinfectar el juego durante 10 minutos antes de que una persona lo use. Cada persona al llegar espera hasta que el empleado le avisa que puede usar el juego, lo usa por un tiempo y luego lo devuelve. Nota: suponga que la persona tiene una función Usar_juego que simula el uso del juego; y el empleado una función Desinfectar_Juego que simula su trabajo. Todos los procesos deben terminar su ejecución.

Solución :

> Acá yo hice un procedimiento más que es esperarLiberacion(), pero hace lo mismo solamente que lo separe. Estaría mal?

```
Monitor Administrador{
    cond espera;
    cond quieroUsarlo;
    int personasEspernado = 0;
    cond yaNoLoUso;

    boolean libre = True;

    procedure llegue(){
        signal(quieroUsarlo);  //Aviso que llegue
        personasEsperando++;
        wait(espera);          //Espero que me habiliten el paso
    }

    procedure devolver(){
        signal(yaNoLoUso) //Le aviso al empleado que ya termine de usar el
juego
    }

    procedure esperarLiberacion(){
        wait(yaNoLoUso);       //Espera a que no lo usen más
    }

    procedure sig(){
        if (personasEsperando > 0) {  //Si hay personas esperando
            personaEsperando--;
            signal(espera);    //Hace pasar a la persona que esta esperando
        }else{
            wait(quieroUsarlo); //Espera a que llegue alguien
        }
    }
}
```

```
process personas[id : 1..N]{
    Administrador.llegue();
    Usar_Juego();
    Administrador.devolver();
}

process empleado {
    for i : 1 to N
        Desinfectar_Juego();
        Delay(10);
        Administrador.sig();
        Administrador.esperarLiberacion();  //Espera a que se libere
}
```

> Esta solución la hice sin passing the condition este si es !!!

```
Monitor Administrador{
    cond espera;
    cond quieroUsarlo;
    int personasEspernado = 0;
    cond yaNoLoUso;

    procedure llegue(){
        signal(quieroUsarlo);    //Aviso que llegue y quiero usarlo
        personasEsperando ++;
        wait(espera);            //Espero a mi turno
    }

    procedure devolver(){
        signal(yaNoLoUso) //Le aviso al empleado que ya termine de usar el
juego
        personasEsperando--;
    }

    procedure esperarLiberacion(){
        wait(yaNoLoUso);       //Espera a que no lo usen más
    }

    procedure sig(){
        if (personasEsperando > 0) {
            signal(espera);    //Hace pasar a la persona que esta esperando
        }else {
```

```
            wait(quieroUsarlo); //Sino espero a que me avisen
        }
    }
}

process personas[id : 1..N]{
    Administrador.llegue();
    Usar_Juego();
    Administrador.devolver();
}

process empleado {
    for i : 1 to N
        Desinfectar_Juego();
        delay(10)
        Administrador.sig();
        Administrador.esperarLiberacion();
}
```

```
Monitor Administrador{
    cond espera;
    cond quieroUsarlo;
    int personasEspernado = 0;
    cond yaNoLoUso;
    boolean estaLimpoa = false; // Cuando esta limpia es que puede pasar
alguien
    cond findLimpieza;

    procedure llegue(){
        personasEsperando ++;
        signal(quieroUsarlo);    //Aviso que llegue y quiero usarlo
        wait(espera);            //Espero a mi turno
    }

    procedure devolver(){
        signal(yaNoLoUso) //Le aviso al empleado que ya termine de usar el
juego

    }

    procedure sig(){}
        if (personasEsperando = 0) { //Hay alguien esperando
            wait(quieroUsarlo); //Sino espero a que me avisen
        }
```

```
        //Esto debe ir afuera del if ya que se despierta por dos formas, que
no haya nadie o que si.
        personasEsperando--;
        signal(espera);    //Hace pasar a la persona que esta esperando
        wait(yaNoLoUso);       //Espera a que no lo usen más
    }
}

process personas[id : 1..N]{
    Administrador.llegue();
    Usar_Juego();
    Administrador.devolver();
}

process empleado {
    for i : 1 to N
        Desinfectar_Juego();
        delay(10);
        Administrador.sig();
}
```