Práctica 2

General: hacer P de lo más restrictivo primero

1. Existen N personas que deben ser chequeadas por un detector de metales antes de poder
ingresar al avión.
a. Analice el problema y defina qué procesos, recursos y semáforos/sincronizaciones
serán necesarios/convenientes para resolverlo.

   - N procesos persona
   - 1 recurso detector de metales
   - Sincronización por exclusión mútua
   - 1 semáforo para indicar que el detector está libre

   b. Implemente una solución que modele el acceso de las personas a un detector (es decir,
si el detector está libre la persona lo puede utilizar; en caso contrario, debe esperar).

```
sem libre=1

process persona[id: 0.. N-1]{
    P(libre);
    //persona accede al detector y es chequeada
    V(libre)
}
```

   c. Modifique su solución para el caso que haya tres detectores.

```
sem libre=3
```

```
process persona[id:0..N-1]{
    P(libre);
    //persona accede al detector y es chequeada
    V(libre);
}
```

d. Modifique la solución anterior para el caso en que cada persona pueda pasar más de
una vez, siendo aleatoria esa cantidad de veces.

```
sem libre=3;

process persona[id:0..N-1]{
    int cant= random;
    while(random> 0) {
        P(libre);
        //persona accede al detector y es chequeada
        V(libre);
        cant:= cant -1
        }
    }
```

2. Un sistema de control cuenta con 4 procesos que realizan chequeos en forma
colaborativa. Para ello, reciben el historial de fallos del día anterior (por simplicidad, de
tamaño N). De cada fallo, se conoce su número de identificación (ID) y su nivel de
gravedad (0=bajo, 1=intermedio, 2=alto, 3=crítico). Resuelva considerando las siguientes
situaciones:
a) Se debe imprimir en pantalla los ID de todos los errores críticos (no importa el
orden).

```
vector documentos[N];
process chequeador[id: 0..3]{
     for i:= id* (N/4) to (id * N/4 + N/4) - 1
         if (documentos[i].nivel==3)
                writeln(documentos[i].id);
}
```

b) Se debe calcular la cantidad de fallos por nivel de gravedad, debiendo quedar los
resultados en un vector global.

```
vector documentos[N];
vector contadorNivel[4];
sem contadorLibre[4] = ([4] 1);

process chequeador[id:0..3]{
    vector contadorLocal]
    for i:= id* (N/4) to (id * N/4 + N/4) - 1{
        contadorLocal[documentos[i].nivel]+=1;
    }

    for i:= 0 to 3{
        P[documentos[i].nivel]
        contadorNivel[documentos[i].nivel]+= contadorLocal
        V[documentos[i].nivel]

}
```

c) Ídem b) pero cada proceso debe ocuparse de contar los fallos de un nivel de
gravedad determinado.

```
vector contadorNivel[4];
```

```
process chequeador[id:0..3]{
    for i:= 1 to N
        if [documentos[i].nivel == id]{
            contadorNivel[id]+=1
        }
}
```

3. Un sistema operativo mantiene 5 instancias de un recurso almacenadas en una cola.
   Además, existen P procesos que necesitan usar una instancia del recurso. Para eso, deben
   sacar la instancia de la cola antes de usarla. Una vez usada, la instancia debe ser encolada
   nuevamente para su reúso.

```
cola recursos;
sem mutex = 1;
sem disponibles= 5;

process procesos[id:0..P-1]{
    elem elemento;
    P(disponibles)
    P(mutex);
    pop(cola, elemento);
    V(mutex);
    //usa el elemento
    P(mutex)
    push(cola, elemento);
    V(mutex);
    V(disponibles)
```

```
}
```

4. Suponga que existe una BD que puede ser accedida por 6 usuarios como máximo al
mismo tiempo. Además, los usuarios se clasifican como usuarios de prioridad alta y
usuarios de prioridad baja. Por último, la BD tiene la siguiente restricción:
• no puede haber más de 4 usuarios con prioridad alta al mismo tiempo usando la BD.
• no puede haber más de 5 usuarios con prioridad baja al mismo tiempo usando la BD.
Indique si la solución presentada es la más adecuada. Justifique la respuesta.

<table>
<tr><td colspan="2">Var
total: sem := 6;
alta: sem := 4;
baja: sem := 5;</td></tr>
<tr><td>Process Usuario-Alta [I:1..L]::
{ P (total);
P (alta);
//usa la BD
V(total);
V(alta);
}</td><td>Process Usuario-Baja [I:1..K]::
{ P (total);
P (baja);
//usa la BD
V(total);
V(baja);
}</td></tr>
</table>

Para mí el orden de los semáforos tanto en los procesos de usuarios de alta y baja prioridad debería ser al revés en la operación P, ya que como está planteado, al hacer primero el P(total) estaría ocupando un espacio que tal vez no estaría ocupando realmente ya que podría quedar demorado en P(alta) o P(baja) en el caso de que ya hubiera 4 o 5 usuarios de alta o baja prioridad respectivamente. De esa forma estaría ocupando un espacio innecesario que

tal vez podría ser ocupado por un proceso cuya prioridad todavía no llegó al máximo permitido. Luego, el orden de las operaciones V esta bien porque libero un lugar .

```
Var
total: sem := 6;
alta: sem := 4;
baja: sem := 5;

Process Usuario-Alta [I:1..L]::
 {
     P (total);
     P (alta);
     //usa la BD
     V(alta);
     V(total);
 }

 Process Usuario-Baja [I:1..K]::
 {
     P (baja);
     P (total);
    //usa la BD
     V(total);
     V(baja);
 }
```

5. 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.

```
vector contenedor[N];
int ocupado = 0;
int libre = 0;
sem lleno = 0;
sem vacío = N;

process Preparador {
    while (true) {
        //preparar paquete
        P(vacío)
        contenedor [libre] = //dejarPaquete
        libre = (libre + 1 MOD N)
        V(lleno);
    }
}

process Entegador {
    while (true) {
        P(lleno);
        //paquete = contenedor [ ocupado ]
        ocupado = ocupado +1 MOD N
        V(vacío);
        //realizar entrega de paquete
    }
}
```

b) Modifique la solución a) para el caso en que haya P empleados Preparadores.

```
vector contenedor[N];
int ocupado = 0;
int libre = 0;
sem lleno = 0;
sem vacío = N;
sem mutexP=1

process Preparador {
    while (true) {
        //preparar paquete
        P(vacío)
        P(mutexP)
        contenedor [libre] = //dejarPaquete
        libre = (libre + 1 MOD N)
        V(mutexP)
        V(lleno);
    }
}

process Entegador {
    while (true) {
        P(lleno);
        //paquete = contenedor [ ocupado ]
        ocupado = ocupado +1 MOD N
        V(vacío);
        //realizar entrega de paquete
    }
}
```

c) Modifique la solución a) para el caso en que haya E empleados Entregadores.

```
vector contenedor[N];
int ocupado = 0;
int libre = 0;
sem lleno = 0;
sem vacío = N;
sem mutexE=1;

process Preparador {
    while (true) {
        //preparar paquete
        P(vacío)
        contenedor [libre] = //dejarPaquete
        libre = (libre + 1 MOD N)
        V(lleno);
    }
}

process Entegador {
    while (true) {
        P(lleno);
        P(mutexE)
        //paquete = contenedor [ ocupado ]
        ocupado = ocupado +1 MOD N
        V(mutexE)
        V(vacío);
        //realizar entrega de paquete
    }
}
```

d) Modifique la solución a) para el caso en que haya P empleados Preparadores y E
empleadores Entregadores.

```
vector contenedor[N];
int ocupado = 0;
int libre = 0;
sem lleno = 0;
sem vacío = N;
sem mutexP=1
sem mutexE=1;

process Preparador {
    while (true) {
        //preparar paquete
        P(vacío)
        P(mutexP)
        contenedor [libre] = //dejarPaquete
        libre = (libre + 1 MOD N)
        V(mutexP)
        V(lleno);
    }
}

process Entegador {
    while (true) {
        P(lleno);
        P(mutexE)
        //paquete = contenedor [ ocupado ]
        ocupado = ocupado +1 MOD N
        V(mutexE)
        V(vacío);
        //realizar entrega de paquete
    }
}
```

6. Existen N personas que deben imprimir un trabajo cada una. Resolver cada ítem usando
   semáforos:

a) 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, sin importar
el orden. Existe una función Imprimir(documento) llamada por la persona que simula el
uso de la impresora. Sólo se deben usar los procesos que representan a las Personas.

```
sem libre=1;

process persona[id: 0.. N-1] {
    P(libre);
    id.imprimir(documento);
    V(libre);
}
```

b) Modifique la solución de (a) para el caso en que se deba respetar el orden de llegada.

```
sem libre = 1;
cola c;
boolean ocupada=False
sem mutex = 1
sem sig[N] = ([N] 0)

process persona[id: 0.. N-1]{
    P(mutex);
    if (! ocupada ){
        ocupada=True
        id.imprimir(documento)
        V(mutex);
    }
    else {
```

```
        c.push(id);
        V(mutex);
        P(sig[id]);
        id.imprimir(documento);
        P(mutex);
        if (c.vacia())
            ocupada=False
        else
            V(sig[c.pop()])
        V(mutex);
        }
    }
```

c) 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 impresora
hasta que no haya terminado de usarla la persona X-1).

```
int sigID =0
sem sig[N]= ([N] 0 )


process Persona [id: 0.. N-1] {
if (id != sigID ){
P(sig[id]);
//usar fotocopiadora
}
else{
//usar fotocopiadora
sigID+=1;
V(sig[sigID]);
}
}
```

d) Modifique la solución de (b) para el caso en que además hay un proceso Coordinador
que le indica a cada persona que es su turno de usar la impresora.

```
sem libre = 1;
cola c;
sem mutex = 1
sem sig[N] = ([N] 0)
sem llena=0;

process persona[id: 0.. N-1]{
    P(mutex);
    c.push(id);
    V(llena)
    V(mutex);
    P(sig[id])
    id.imprimir(documento);
    V(libre);
    }

    process Coordinador(){
        while(true){
            P(llena);
            P(mutex);
            V(sig[c.pop])
            V(mutex);
            P(libre);
        }
    }
```

e) Modificar la solución (d) para el caso en que sean 5 impresoras. El coordinador le
indica a la persona cuando puede usar una impresora, y cual debe usar.

```
sem libre = 5;
cola c;
sem mutex = 1
vector impresora[5]
int pos=0
sem sig[N] = ([N] 0)
sem llena=0;
int idImpresora[N]
int posLibre=5;
sem mutexpos=1;
process persona[id: 0.. N-1]{
    P(mutex);
    c.push(id);
    V(llena)
    V(mutex);
    P(sig[id])
    id.impresora[idImpresora[id]](documento);
    //imprime
    P(mutexPos);
    posLibre+=1
    V(mutexPos);
    V(libre);

}

process Coordinador(){
    int siguiente;
        while(true){
            P(llena)
            P(libre)
            posLibre-=1
            P(mutex);
            siguiente=c.pop
```

```
            V(mutex);
            idImpresora[siguiente]= posLibre
            V(sig[siguiente])
            P(mutexPos);
            posLibre -=1;
            V(mutexPos);
        }
    }
```

7. 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).

```
sem mutex=1;
int cantidad=0;
sem barrera=0;
sem puntaje [10]= ([10] 0);
vector enunciado[10]; // posee un campo para la cantidad que
cola tareas;
sem termine=0;
```

```
process Alumno [id: 0.. 49] {
    int tarea;
    tarea= elegir();
    int nota;
    P(mutex)
    cantidad +=1;
    if ( cantidad = 50 )
        for i:= 1 to 50
            V(barrera);
    else
        P(barrera);
    //hacer tarea
    P(mutex);
    tareas.push(tarea);
    V(mutex);
    V(termine);
    P(puntaje[tarea]);
    nota:= enunciado[tarea].puntaje;
}

process Profesor(){
    nota:=10;
    for i:= 1 to 50{
        P(termine);
        P(mutex);
        tarea:= tareas.pop();
        V(mutex);
        enunciado[tarea].total+=1;
        total:= enunciado[tarea].total;
        if (total == 5 ){
            enunciado[tarea].puntaje= nota;
            nota-=1;
            for i:= 1 to 5
                V(puntaje[tarea]
        }
```

```
    }
}
```

8. 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.
a) Implemente una solución asumiendo que T > E.

```
sem barrera=0
sem mutex=1
int contador=0
sem producto= T
cola c;
vector contadorEmpleado [E]
process empleado[id=0.. E-1]{
    P(mutex)
    contador+=1
    if (contador  == E )
        for i:= 1 to E
            V(barrera)
    V(mutex)
    P(barrera)
    P(mutex)
    while(!c.vacia()){
        pieza= c.pop
```

```
        V(mutex)
        pieza.producir
        contadorEmpleado[id]+=1
        P(mutex)
    }
     V(mutex)
}
```

b) Implemente una solución que contemple cualquier valor de T y E.

```
sem barrera=0
sem mutex=1
int contador=0
sem producto= T
cola c;
vector contadorEmpleado [E]
process empleado[id=0.. E-1]{

    int i;
    P(mutex)
    contador+=1
    if(T<=E &&  contador  == T )//pensando en que no hace
        for i:= 1 to T
            V(barrera)
    else
        if (T> E) && contador==E)
            for i:= 1 to E
              V(barrera)
    V(mutex)
    P(barrera)
    P(mutex)
    while(!c.vacia()){
        pieza= c.pop
        V(mutex)
        pieza.producir
        contadorEmpleado[id]+=1
```

```
        P(mutex)
    }
     V(mutex)
}
```

9. 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).

```
sem Mdepositado= 0;
int cantMarcos=0;
sem Vdepositado=0;
int cantVidrios=0;
cola colaMarco;
cola colaVidrio;
sem lugarMarcos=30;//30
sem lugarVidrios=30;
sem mutexM=1;
sem mutexArmador=1;

process carpintero[id:0..3]{
     elem marco;
     while (true){
```

```
            marco= hacerMarco();
            P(lugarMarcos)
            P(mutexM)
            colaMarco.push(marco);
            V(Mdepositado)
              V(mutexM)
        }
}

process vidriero(){
    boolean lugar=True
    elem vidrio
    while (true){
        vidrio = hacerVidrio()
        P(lugarVidrios)
        P(mutexV)
        colaVidrio.push(vidrio);
        V(Vdepositado)
        V(mutexV)
    }
}

process armador[id :0..1]{
    elem vidrio, marco, ventana;
    while(true){
        P(mutexArmador);
        P(Vdepositado);
        P(Mdepositado);
        P(mutexM);
        marco= colaMarco.pop()
        V(lugarMarcos);
        V(mutexM);
        P(mutexV);
        vidrio = pop.colaVidrio()
        V(lugarVidrios);
        V(mutexV);
        V(mutexArmador);
        ventana= hacerVentana(marco, vidrio);
```

```
    }
}
```

10. A una cerealera van T camiones a descargarse trigo y M camiones a descargar maíz. Sólo
hay lugar para que 7 camiones a la vez descarguen, pero no pueden ser más de 5 del mismo
tipo de cereal.
a) Implemente una solución que use un proceso extra que actúe como coordinador
entre los camiones. El coordinador debe retirarse cuando todos los camiones han
descargado. //cola en la que se encolen sin importar el tipo

b) Implemente una solución que no use procesos adicionales (sólo camiones).

```
sem maíz =5;
sem total= 7;
sem trigo=5;
process camionTrigo [ id: 0..T-1]{
    P(trigo)
    P(total)
    //descargarTrigo
    V(total)
    V(trigo)
}
```

```
process camionMaiz[id: 0.. M-1]{
    P(maiz)
    P(total)
    //descargarMaiz
    V(total)
    V(maiz)
}
```

11.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; suponga que el empleado tienen una
función VacunarPersona() que simula que el empleado está vacunando a UNA persona.

```
sem mutex=1;
cola c;
int total=0;
sem empezar= 0;
sem contador=1;
sem atendida[50]= ([50] 0)

process empleado(){
    int atendidas=0;
    persona p;
    while (atendidas != 50){
        P(empezar)
        for i:= 1 to 5
            P(mutex)
```

```
            p= c.pop()
            V(mutex);
            VacunarPersona()
            V(atendida[p])
            atendidas+=1
    }
}


process persona[id:0.. 49]{
        P(mutex)
        c.push(id)
        V(llegue)
        V(mutex)
        P(contador);
        total+=1;
        if (total == 5){
            total=0
            V(empezar);
        }
        V(contador);
        P(atendida [id ])
}
```

12.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.

```
vector cantidadPuesto [3]
sem contadorSem[3]=([3]1)

sem procesos[150]= ([150] 0)
vector vectorCola[3] c1, c2, c3; //vector de colas
sem mutex[3]=([3] 1)
sem recepcionista=1;
vector pasajeros[150];
sem contador=1;
sem llegueE[3]= ([3]0)
sem llegueP=0;
cola general;
sem mutexC=1;
process Pasajeros [id: 0..149]{
    int miPuesto
    P(mutexC)
    general.push(id)
    V(mutexC)
    V(llegueP)
    miPuesto=pasajeros[id]
    P(mutex[miPuesto]) //cola del puesto
    vectorCola[miPuesto].push(id)
    V(llegueE[miPuesto])
    V(mutex[miPuesto])
    P(procesos[id])
    //hisopando
    //retirado
}



process Recepcionista(){
    int minimo, id;
```

```
    int tot=0;
    while (tot!=150){
        P(llegueP)
        P(mutexC)
        minimo= calcularMinimo()
        id= general.pop()
        V(mutexC)
        pasajeros[id] = minimo

        P(contadorSem[minimo])
        cantidadPuesto[minimo]+=1;
        V(contadorSem[minimo])
        tot+=1;
    }
}


process Enfermera(id: 0..2){
    while(true){
        P(llegueE[id])
        P(mutex[id])//cola del puesto
        paciente= vectorCola[id].pop()
        V(mutex[id])
        hisopar(paciente)
        V(procesos[paciente]);
        P(contadorSem[id]);
        cantidadPuesto[id]-=1;
        V(contadorSem[id]);
     }

}
```

b) Modifique la solución anterior para que sólo haya procesos Pasajeros y Enfermera,
siendo los pasajeros quienes determinan por su cuenta qué puesto tiene

menos
personas esperando.

Cuando calculo el minimo bloqueo el vector contador

```
vector cantidadPuesto [3];
sem procesos[150]= ([150] 0)
vector vectorCola[3] c1, c2, c3; //vector de colas
sem mutex[3]=([3] 1)
sem contadorSem[3]=([3]1)
vector pasajeros[150];
sem contador=1;
sem llegueE[3]= ([3]0)
sem llegueP=0;

process Pasajeros [id: 0..149]{
    int miPuesto
    V(llegueP)
    for i:= 1 to 3
            P(contadorSem[i])
    miPuesto= calcularMinimo(cantidadPuesto)
    for i:= 1 to 3
        V(contadorSem[i])
    P(mutex[miPuesto]) //cola del puesto
    vectorCola[miPuesto].push(id)
    V(llegueE[miPuesto])
    V(mutex[miPuesto])
    P(procesos[id])
    //hisopando
    //retirado
}



process Enfermera(id: 0..2){
    while(true){
        P(llegueE[id])
        P(mutex[id])//cola del puesto
```

```
        paciente= vectorCola[id].pop()
        V(mutex[id])
        hisopar(paciente)
        V(procesos[paciente]);
        P(contadorSem[id]);
        cantidadPuesto[id]-=1;
        V(contadorSem[id]);
    }

}
```

Nota: suponga que existe una función Hisopar() que simula la atención del pasajero por
parte de la enfermera correspondiente.