Práctica 3 – Monitores

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?

a) Funciona correctamente, a partir del segundo proceso que entre a **entrarPuente()** va a dormir el proceso hasta que cant vuelva a valer 0. Lo único incorrecto es el llamado dentro de **Process Auto** a los procedimientos de **Puente** ya que pasa como parametro el auto cuando no corresponde.

b)

```
Monitor Puente
   Procedure cruzarPuente(){
      cruzar();
   }
End Monitor;

Process Auto [a:1..M]
   Puente.cruzarPuente();
End Process;
```

c) La primer solución respeta el orden de llegada, la segunda solución no respeta el orden de llegada.

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.

```
Monitor BD
   int enUso = 0;
   cond cola;

   Procedure solicitarAcceso(){
      if (enUso < 5){
         enUso++;
```

```
      }
      else{
         wait(cola);
         enUso++;
      }
   }

   Procedure marcarSalida(){
      enUso--;
      signal(cola);
   }
End Monitor;

Process lector [l:1..N]{
   BD.solicitarAcceso();
   UsarBD;
   BD.marcarSalida();
}
```

3. 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.

a)

```
Monitor Fotocopiadora
   Procedure usarFotocopiadora(){
      Fotocopiar();
   }
End Monitor;

Process Persona [p:1..N]
   Fotocopiadora.usarFotocopiadora();
End Process;
```

b)

```
Monitor Fotocopiadora
   cond cola;
```

```
   int enEspera = 0;
   bool libre = true;
   Procedure entrar(){
      if (not libre){
         enEspera++;
         wait(cola);
      }
      else {libre = false}
   }

   Procedure salir(){
      if (enEspera > 0){
         enEspera--;
         signal(cola);
      }
      else {libre = true}
   }
End Monitor;

Process Persona [p:1..N]
   Fotocopiadora.entrar();
   Fotocopiar();
   Fotocopiadora.salir();
End Process;
```

c)

```
Monitor Fotocopiadora
   cond cola[N];
   Cola colaEspecial[N];
   int enEspera = 0;
   bool libre = true;
   Procedure entrar(in int id,in int edad){
      if (not libre){
         enEspera++;
         colaEspecial.push(id,edad);
         wait(cola[id]);
      }
      else {libre = false}
   }

   Procedure salir(){
      int idAux = 0;
      if (enEspera > 0){
         enEspera--;
         idAux = colaEspecial.pop();
         signal(cola[idAux]);
      }
      else {libre = true}
   }
End Monitor;

Process Persona [p:1..N]
   int edad = alguna edad;
   Fotocopiadora.entrar(p,edad);
```

```
   Fotocopiar();
   Fotocopiadora.salir();
End Process;
```

d)

```
Monitor Fotocopiadora
   int turno = 1;
   cond cola[N];
   Procedure entrar(in int id){
      if (turno != id){
         wait(cola[id]);
      }

   Procedure salir(){
      turno++;
      signal(cola[turno]);
   }
End Monitor;

Process Persona [p:1..N]
   Fotocopiadora.entrar(p);
   Fotocopiar();
   Fotocopiadora.salir();
End Process;
```

e)

```
Monitor Fotocopiadora
   cond cola;
   cond empleado;
   cond fin;
   int enEspera = 0;
   bool libre = true;
   Procedure entrar(){
      signal(empleado);
      enEspera++;
      wait(cola);
   }

   Procedure salir(){
      signal(fin);
   }

   Procedure asignar(){
      if (enEspera == 0){
         wait(empleado);
      }
      enEspera--;
      signal(cola);
      wait(fin);
   }
End Monitor;
```

```
Process Persona [p:1..N]
   Fotocopiadora.entrar();
   Fotocopiar();
   Fotocopiadora.salir();
End Process;
Process Empleado
   for (int i=0 to N){
      Fotocopiadora.asignar();
   }
End Process;
```

f) Para otro momento.

4. 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).

```
Monitor Puente
   cond cola;
   cond afuera;
   int enEspera = 0;
   int pesoTotal = 0;
   int enEsperaAfuera = 0;
   bool libre = true;

   Procedure entrar(int in peso){
      if (not libre){
         enEspera++;
         wait(cola);
      }
      else {
         libre = false;
      }
      if (pesoTotal + peso > 50000){
         wait(afuera);
      }
      pesoTotal += peso;
   }

   Procedure salir(int  in peso){
      pesoTotal -= peso;
      if (enEsperaAfuera > 0){
         enEsperaAfuera--;
         signal(afuera);
      }
      if (enEspera > 0){
         enEspera--;
         signal(cola);
      }
      else {libre = true}
   }
End Monitor;
```

```
Process Vehiculo [v:1..N]
   Puente.entrar(v.peso);
   Cruzar();
   Puente.salir(v.peso);
End Process;
```

A PARTIR DE ESTE PUNTO USE MarkDown (MD).

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.