

**Posibles soluciones a los ejercicios del parcial práctico del 13-12-22**

*Importante: las soluciones que se muestran a continuación no son las únicas que se pueden considerar correctas para los dos ejercicios planteados.*

1. Resolver con *SEMÁFOROS* el siguiente problema. 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.

```
int empleados = 0;
int grupos = 0;
int muebles[10] = [10]{0};
sem s_reunion[10] = [10]{0};
sem s_muebles[10] = [10]{1};
sem s_llegada = 1;

Process Empleado [i=0..49] {
    int mi_grupo;
    // sincronizar llegada
    P(s_llegada);
    mi_grupo = grupos;
    empleados++;
    if (empleados < 5) {
        V(s_llegada);
        P(s_reunion[mi_grupo]);
    } else {
        for i= 1 to 4 do
            V(s_reunion[mi_grupo]);
        empleados = 0;
        grupos++;
        V(s_llegada);
    }
    // trabajar
    P(s_muebles[mi_grupo]);
    While (muebles[mi_grupo] < M) {
        muebles[mi_grupo]++;
        V(s_muebles[mi_grupo]);
        // armar mueble
        sleep(rand());
        P(s_muebles[mi_grupo]);
    }
    V(s_muebles[mi_grupo]);
}
```

2. Resolver con **MONITORES** el siguiente problema. En un comedor estudiantil hay un horno microondas que debe ser usado por *E estudiantes* de acuerdo con el orden de llegada. Cuando el estudiante accede al horno, lo usa y luego se retira para dejar al siguiente. *Nota:* cada Estudiante una sólo una vez el horno; los únicos procesos que se pueden usar son los “*estudiantes*”.

```
Monitor Horno {
    int usando = 0, esperando=0;
    cond cola;

    procedure pedir () {
        if (usando > 0){
            esperando++;
            wait(cola);
        }
        else
            usando++;
    }

    procedure liberar() {
        if (esperando > 0){
            signal(cola);
            esperando--;
        } else
            usando--;
    }
}

Process Estudiante [i: 1..E]{
    Horno.pedir();
    // Calentar
    sleep (rand());
    Horno.liberar();
}
```

1. Resolver con **Pasaje de Mensajes Sincrónicos (PMS)** el siguiente problema. En un comedor estudiantil hay un horno microondas que debe ser usado por *E estudiantes* de acuerdo con el orden de llegada. Cuando el estudiante accede al horno, lo usa y luego se retira para dejar al siguiente. *Nota:* cada Estudiante una sólo una vez el horno.

```
Process Estudiante [i=0..E-1] {  
  
    Admin ! pedido (i);  
    Admin ? usar ();  
    // usar el horno  
    Admin ! liberar ();  
}  
  
Process Admin {  
  
    bool libre = true;  
    queue estudiantes;  
    int idEst;  
  
    while (true) {  
        If (libre = true); Estudiante [*] ? pedido (idEst) ->  
            libre = false;  
            Estudiante[idEst] ! usar ();  
        [] (libre = false); Estudiante [*] ? pedido (idEst) ->  
            q_push(estudiantes,idEst);  
        [] Estudiante [*] ? liberar (idEst) ->  
            if (empty(estudiantes))  
                libre = true;  
            else  
                Estudiante [pop(estudiantes)] ! usar ();  
    }  
}
```

2. Resolver con **ADA** el siguiente problema. Se debe controlar el acceso a una base de datos. Existen *L procesos Lectores* y *E procesos Escritores* que trabajan indefinidamente de la siguiente manera:

- *Escritor*: intenta acceder para escribir, si no lo logra inmediatamente, espera 1 minuto y vuelve a intentarlo de la misma manera.
- *Lector*: intenta acceder para leer, si no lo logra en 2 minutos, espera 5 minutos y vuelve a intentarlo de la misma manera.

Un proceso Escritor podrá acceder si no hay ningún otro proceso usando la base de datos; al acceder escribe y sale de la BD. Un proceso Lector podrá acceder si no hay procesos Escritores usando la base de datos; al acceder lee y sale de la BD. Siempre se le debe dar prioridad al pedido de acceso para escribir sobre el pedido de acceso para leer.

```
TASK Type ProcesoLector;  
TASK Type ProcesoEscritor;  
  
TASK Type AdministradorDeBD IS  
    entry entrada_lector ();  
    entry entrada_escritor ();  
    entry salida_lector();  
    entry salida_escritor();  
end AdministradorDeBD;  
  
admin: AdministradorDeBD;  
procesos_lectores: array (1..L) of ProcesoLector;  
procesos_escritores: array (1..E) of ProcesoEscritor;  
  
TASK Body ProcesoLector IS  
Begin  
    loop  
        SELECT  
            admin.entrada_lector();  
            -- Leer  
            admin.salida_lector();  
        OR DELAY 120.0  
            DELAY (5*60);  
        End SELECT;  
    End loop;  
End ProcesoLector;
```

```
TASK Body ProcesoEscritor IS
Begin
  loop
    SELECT
      admin.entrada_escritor();
      -- Escribir
      admin.salida_escritor();
    ELSE
      DELAY (60)
    End SELECT;
  End loop;
End ProcesoEscritor;
```

```
TASK Body AdministradorDeBD IS
  lectores: int = 0;
Begin
  loop
    SELECT
      WHEN (lectores == 0) =>
        Accept entrada_escritor ();
        -- espera a que termine el escritor
        Accept salida_escritor();
      OR
      WHEN (entrada_escritor'count == 0) =>
        Accept entrada_lector ();
        lectores++;
      OR
        Accept salida_lector ();
        lectores--;
      End SELECT;
    End loop;
  End AdministradorDeBD;
```