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 logro 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;
```