3. *(3 p)* Una persona tiene en su casa una jaula llena de canarios en la que hay un plato de alpiste y un columpio. Todos los canarios quieren primero comer del plato y luego columpiarse, sin embargo sólo tres de ellos pueden comer del plato al mismo tiempo y solo uno de ellos puede columpiarse. Escribir el pseudocódigo basado en C de un programa que usando un **monitor** de nombre `jaula` coordine la actividad de los canarios. Considerar la solución de Hansen en el comportamiento de la operación `signal_mon`.

En la Figura 1 se muestra una posible solución para este ejercicio. Esta solución utiliza las siguientes variables globales y variables de condición:

- `contadorP`. Variable global de tipo entero para llevar la cuenta del número de puestos en el plato ocupados.
- `contadorC`. Variable global de tipo entero para indicar si el columpio está ocupado.
- `puesto_plato_disponible`. Variable de condición para bloquear en su cola a los procesos hasta que exista algún puesto en el plato disponible.
- `columpio_disponible`. Variable de condición para bloquear en su cola a los procesos hasta que el columpio esté disponible.

Señalar que las acciones de `comer()` y `columpiarse()` deben invocarse (tanto en ésta como en cualquier otra solución) dentro del código del proceso `canario()` para garantizar la máxima concurrencia de procesos. Si se invocaran dentro de procedimientos del monitor, como en un monitor solo se puede ejecutar un procedimiento a la vez, se tendría la situación de que no podrían comer tres canarios y columpiarse otro simultáneamente, ya que mientras un canario come no podría comer ni columpiarse ningún otro, o mientras un canario se columpia no podrían comer los demás.

```
#define N 3 /* Número de  puestos en el plato */
monitor jaula /* Definición del monitor */
    condición puesto_plato_disponible, columpio_disponible;
    int contadorP, contadorC;

    void obtener_puesto_en_plato()  /* Procedimiento del monitor */
    {
      if (contadorP == N) wait_mon(puesto_plato_disponible);
      contadorP=contadorP+1;
    }

    void dejar_puesto_plato() /* Procedimiento del monitor */
    {
      contadorP = contadorP - 1;
      signal_mon(puesto_plato_disponible);
    }

    void obtener_columpio() /* Procedimiento del monitor */
    {
      if (contadorC == 1) wait_mon(columpio_disponible);
      contadorC=contadorC+1;
    }

    void dejar_columpio() /* Procedimiento del monitor */
    {
      contadorC = contadorC - 1;
      signal_mon(columpio_disponible);
    }

    { /* Inicialización del monitor */
      contadorP=0, contadorC=0;
    }
end monitor

void canario() /* Proceso canario */
{
    jaula.obtener_puesto_plato();
    comer();
    jaula.dejar_puesto_plato();

    jaula.obtener_columpio();
    columpiarse();
    jaula.dejar_columpio();
}

main() /* Ejecución concurrente */
{
    ejecución_concurrente(canario,...,canario);
}
```