UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA
Escuela de Ingeniería Informática

Fundamentos de los Sistemas Operativos
Examen parcial – 11 de abril de 2019

1	2	3	test	extra	NOTA

Nombre y apellidos	**DNI/NIE**

DURACIÓN: *Dispone de dos horas para realizar el examen.*

Lea las instrucciones para el test en la hoja correspondiente.

1 **(1 punto)** A un planificador de CPU llegan cuatro procesos, según el cuadro de la derecha. Para estas dos políticas: SJF expulsivo y Round Robin (Q=2), obtenga lo siguiente:

- Diagrama de Gantt con la planificación.
- Tiempo de espera de cada uno de los procesos y valor medio del conjunto.
- Número de cambios de contexto realizados durante la planificación.

proceso	llegada	duración
A	0	6
B	2	1
C	4	1
D	5	2

2 **(1'25 puntos)** Para cada una de estas operaciones, señale cuáles deben ser privilegiadas y por qué. Utilice un máximo de 80 palabras para cada apartado.

1. Modificar la cola de preparados.
2. Volcar en disco el contenido de la memoria RAM.
3. Reprogramar el temporizador del sistema.
4. Leer el contenido del vector de interrupciones.
5. Modificar el valor del registro base de memoria.

```
1  const int N = ...;
2  class Zapato { ... }
3  class Cesto {
4    Cesto (int capacidad);
5    void meterZapatoDerecho(Zapato);
6    void meterZapatoIzquierdo(Zapato);
7    Zapato sacarZapatoDerecho();
8    Zapato sacarZapatoIzquierdo();
9  }

10 Cesto cesto = new Cesto(N);

11 // hilos productores

12 void ObreroDerechos() {
13   while (true) {
14     ... fabricar un zapato derecho Z
15     cesto.meterZapatoDerecho(Z);
16   }
17 }

18 void ObreroIzquierdos() {
19   while (true) {
20     ... fabricar un zapato izquierdo Z
21     cesto.meterZapatoIzquierdo(Z);
22   }
23 }

24 // hilo empaquetador

25 void Empaquetador() {
26   while (true) {
27     Zapato izq,dch;
28     izq = cesto.sacarZapatoIzquierdo();
29     dch = cesto.sacarZapatoDerecho();
30     ... empaquetar zapatos (izq,dch)
31   }
32 }
```

3 (1'75 puntos) Tenemos un sistema concurrente que simula una fábrica de zapatos. En esta fábrica tenemos unos obreros que están constantemente fabricando zapatos de un pie (izquierdo o derecho). Cuando un obrero finaliza un zapato, lo pone en un cesto compartido. Aparte, un obrero empaquetador va sacando del cesto pares de zapatos (uno izquierdo y otro derecho) y los empaqueta.

El cesto tiene una capacidad limitada de N zapatos. Si un fabricante se encuentra el cesto lleno, se tiene que esperar. Si el empaquetador ve el cesto vacío o con zapatos de un solo pie, se tiene que esperar hasta que haya al menos un par completo.

El esquema del código de los procesos se muestra en el cuadro de la izquierda. Tal y como está el código, no hay ninguna protección sobre el uso concurrente del objeto `cesto` y tampoco se comprueban las condiciones de espera. Añada el código que hace falta para resolver esos defectos:

- Utilice unas variables de estado para conocer el estado del sistema.
- Coloque las condiciones de bloqueo/espera donde sea necesario.
- Delimite las zonas del código que deben ejecutarse en exclusión mutua.
- Use las primitivas **ENTRARSC**, **SALIRSC**, **DORMIR** y **DESPERTAR** para ejecutar las acciones de sincronización. Si se siente más cómoda/o con mutex y variables condición, hágalo con ellas (el tipo de herramienta de sincronización utilizada no influirá en la calificación).