Mayo 2025

1. Explica brevemente qué hacen las siguientes líneas de bash, indicando el significado de los argumentos y los separadores. Indica también cuál es su salida.

```
A. #!/bin/bash
B. kill -9 23498
C. read nombre; echo "Hola $nombre y `date`"
D. print="`ls *.dat`"
E. echo $?
F. if [ -d "$1"]
G. head -5 [a-u]?
```

2. Se tienen dos procesos que tienen la variable compartida sum inicializada a 0. No hay proteccion para garantizar la exclusión mutua. Ambos realizan un bucle de N iteraciones donde realizan la operacion sum++ (literalmente) , es la única operacion que realizan sobre la variable. Indica cual sería el lugar más adecuado para introducir un sleep() de forma que se incremente la posibilidad de carrera. Razona la respuesta.

3. Sea un sistema con cinco procesos, P1 a P5, y tres tipos de recursos, A, B y C, de los que existen 10, 5 y 6 ejemplares o instancias, respectivamente. Supongamos que en el instante actual tenemos la situación del sistema dada por las tablas adjuntas. Contesta a las siguientes cuestiones:

```
 A. ¿Cómo actuaría el algoritmo del banquero si ahora, el proceso P2 hace una petición de una instancia del
recurso c?
 B. ¿Y si posteriormente P2 pide una instancia del recurso A?
```

	Recursos asignados			Recursos pendientes		
Proceso	A	B	C	A	B	C
P1	0	1	0	7	4	3
P2	2	0	0	1	2	3
P3	3	0	2	5	0	2
P4	2	1	1	0	1	0
P5	0	0	2	4	3	2

5. Dada el código de los siguientes procesos. Representa la gráfica de trayectorias y explica las conclusiones que se pueden extraer sobre los interbloqueos. Ten en cuenta que los recursos disponibles durante la ejecución R[1], R[2] y R[3].

```
            PROCESO P1                                          PROCESO P2
-------------------------------------------------------------------------------------
    for(i=0; i<2; i++){                                 for(i=0; i<2; i++){
        ops1()                                              ops1()
        obtener(R[i+1])                                     j=i+3; j=(j%3)+(j%4)
        ops2()                                              obtener(R[j])
        obtener(R[i+2])                                     obtener(R[i+2])
        j=i+3; j=(j%3)+(j%4)                                ops4()
        obtener(R[j])                                       obtener(R[i+1])
        ops3()                                              ops3()
        liberar(R[j])                                       liberar(R[j])
        liberar(R[i+2])                                     liberar(R[i+2])
        liberar(R[i+1])                                     liberar(R[i+1])
    }                                                   }
```

6. Dado el siguiente código de la función test del problema de los filósofos, explica el motivo de incluir de up(&s[i]).

```
void probar(i)                  /* i: número de filósofo, de 0 a N−1 */
{
        if (estado[i] == HAMBRIENTO && estado[IZQUIERDO] != COMIENDO && estado[DERECHO] != COMIENDO) {
                estado[i] = COMIENDO;
                up(&s[i]);
        }
}
```