2)

7. Dado el clásico problema de productores y consumidores, se plantea una solución utilizando semáforos con el siguiente código:

```
Typedef int semaforo; .
typedef char* msg;
int N=100; /*Longitud del buffer */
semaforo mutex = 1; /*Da la exclusión mutua */
semaforo lleno = 0; /*Cuenta lugares llenos */
semaforo vacio = N; /*Cuenta lugares vacíos */
```

```
Productor()                    Consumidor()
{                              {
msg mensaje;                   msg mensaje;
while(TRUE)                    while(TRUE)
{                              {
producir(mensaje);             down(&lleno);
down(&mutex);                  down(&mutex);
down(&vacio);                  remover_msg(mensaje);
entrar_msg(mensaje);           up(&mutex);
up(&mutex);                    up(&vacio);
up(&lleno);                    consumir_msg(mensaje);
}                              }
}                              }
```

¿La solución planteada es válida?. En caso de que no lo sea, explique por qué.

8. Se tienen las siguientes secuencias de ejecución:

   a) La secuencia permitida es: ABCDEABCDEABCDEABCDEABCDEABCDE ...

   b) La secuencia permitida es: ACDEBCDEACDEBCDEACDEBCDEACDEBCDE ...

   c) La secuencia permitida es: (A o B)CDE(A o B)CDE(A o B)CDE(A o B)CDE ...

   d) La secuencia permitida es: (AóB)CE(AóB)(AóB)DE(AóB)CE(AóB)(AóB)DE ...

   Realizar la sincronización de los procesos utilizando semáforos para cada uno de los casos especificados.