- Prof. Mark Alan Junho Song
- 812839 - Vinícius Miranda de Araújo

1. **Considere o seguinte código:**

```
n        : integer
s, delay : semaforo // binário

produtor
{
    loop
    {
         wait(s);
         critico 1;
         n = n + 1;
         if (n == 1) then signal(delay);
         signal(s);
    }
}

consumidor
{
    wait(delay);
    loop
    {
        wait(s);
        critico 2;
        n = n - 1;
        if (n == 0) then wait(delay);
        signal(s);
    }
}

// execucao principal
begin
    n     = 0;
    s     = 1;
    delay = 0;
    cobegin
        produtor;
        consumidor;
    coend
end
```

Discuta a correção da solução apresentada.
RESPOSTA:

- Produtor:
  - O produtor entra na região crítica.
  - Aumenta um item `(n = n + 1)`.
  - Se o item produzido foi o primeiro (`n == 1`), libera o consumidor chamando `signal(delay)`.
  - Sai da região crítica.
  - Isso garante que, quando o buffer estava vazio e o consumidor estava bloqueado, ele seja acordado.
- Consumidor:
  - Antes de começar, o consumidor faz `wait(delay)`. Isso o bloqueia até que o produtor coloque ao menos 1 item.
  - Dentro do loop:
    - Entra na região crítica.
    - Retira 1 item (`n = n - 1`).
    - Se o buffer ficou vazio (`n == 0`), ele mesmo faz um `wait(delay)` → volta a bloquear até o produtor repor.
    - Sai da região crítica.
  - Isso garante que o consumidor nunca consuma de um buffer vazio.

2. **Cite as condições necessárias para ocorrência de deadlock. Descreva cada uma em detalhe.**
   RESPOSTA: Um deadlock pode ocorrer se 4 condições existirem ao mesmo tempo, essas são necessárias, mas não suficientes:
   - Exclusão Mútua: apenas um processo por vez pode usar o recurso.
   - Posse e espera: um processo em posse de pelo menos um recurso está esperando por recursos adicionais mantidos por outros processos.
   - Não preempção: um recurso só pode ser liberado voluntariamente pelo processo que o mantém, após ter completado sua tarefa.
   - Espera circular: existe um conjunto $P_0, P_1, \cdots, P_n$ de processos em espera tal que P_0 espera por recurso mantido por P_1, P_1 espera por recurso mantido por $P_2, \cdots, P_{n-1}$ espera por recurso mantido por P_n, e P_n espera por recurso mantido por P_0.
3. **Um computador tem 6 fitas, com N processos competindo pelas mesmas. Cada processo necessita de 2 fitas. Para quais valores de N o sistema é livre de deadlocks?**
   RESPOSTA: Cada processo necessita de 2 fitas. Se houver $N \geq 6$, é possível que cada processo segure 1 fita, ocupando todas as 6 e bloqueando na espera da segunda, o que caracteriza um deadlock. Já para $N \leq 5$, sempre sobra ao menos 1 fita livre, permitindo

que algum processo obtenha sua segunda fita e libere recursos. Logo, o sistema é livre de deadlocks se $N \leq 5$; para $N \geq 6$ o deadlock pode ocorrer.

4. **Considere a situação em que 4 processos A, B, C, D concorrem por recursos da máquina onde existem 2 unidades de fita, 2 unidades de disco e uma unidade de impressão. Os processos se encontram na seguinte situação:**

    1. **O processo A está de posse de uma unidade de fita e requisita uma unidade de disco;**
    2. **O processo B está de posse de uma das unidades de disco;**
    3. **O processo C está de posse da outra unidade de disco e requisita uma unidade de fita;**
    4. **O processo D está de posse da outra unidade de fita;**

    O processo D requisita a unidade de impressão, toma posse do recurso. Logo após processo B faz a mesma requisição. Em seguida D requisita uma unidade de disco. A situação leva a um impasse (deadlock)? Por quê?
    RESPOSTA: Sim, a situação leva a um deadlock. O processo D está de posse da unidade de impressão e requisita um disco, mas todos os discos já estão alocados. O processo B, que possui um disco, requisita a impressão, que está com D. Assim, B espera D liberar a impressão e D espera B liberar o disco, formando um ciclo de espera circular. Portanto, há deadlock.