Quiz 7

O e-mail do participante (**igor.ribeiro@unifesp.br**) foi registrado durante o envio deste formulário.

1. No problema dos produtores-consumidores, os semáforos controlam _____ *

- (●) o número de posições (itens) no buffer
- ( ) o número de produtores produzindo
- ( ) o número de filas vazias e cheias
- ( ) o número de consumidores consumindo

Assinale a(s) alternativa(s) VERDADEIRAS: *

- [ ] O problema do produtor-consumidor é um caso especial do problema dos leitores-escritores em que o produtor é o escritor e o consumidor é o leitor.
- [x] No problema dos leitores-escritores, queremos permitir que vários leitores estejam na seção crítica para os dados compartilhados ao mesmo tempo.
- [ ] No problema dos leitor-escritores, queremos permitir que vários escritores estejam na seções crítica para os dados compartilhados ao mesmo tempo.
- [ ] No problema dos leitores-escritores, queremos permitir que um leitor e um escritor estejam na seção crítica para os dados compartilhados ao mesmo tempo.
- [ ] No problema do produtor-consumidor o produtor nunca fica bloqueado.

No problema do produtor-consumidor indique as assertivas verdadeiras: 1. Assume-se um buffer de n posições, cada um capaz de armazenar um item. 2. O buffer deve ser controlado por exclusão mútua, por exemplo, com um semáforo binário iniciado em 1. 3. O buffer deve ser controlado por exclusão mútua, por exemplo, com um semáforo binário iniciado em 0. 4. Os semáforos para "cheio" e "vazio" contam o número de posições ocupadas e vazias. O semáforo "vazio" é inicializado com o valor de n, enquanto que o semáforo "cheio" é inicializado com o valor 0. 5. Os semáforos para "cheio" e "vazio" contam o número de posições ocupadas e vazias. O semáforo "vazio" é inicializado com o valor 0, enquanto que o semáforo "cheio" é inicializado com o valor de n. *

- ○ 1,3,5
- ◉ 1,2,4
- ○ 1,2,5
- ○ 3,4,5

O problema do jantar dos filósofos ocorre no seguinte caso: *

- ◉ 5 filósofos para 5 garfos
- ○ 4 filósofos para 5 garfos
- ○ 3 filósofos para 5 garfos
- ○ 6 filósofos para 5 garfos

Uma solução sem deadlock para o problema do jantar dos filósofos _____ *

- ○ necessariamente elimina a possibilidade de starvation
- ◉ não elimina necessariamente a possibilidade de starvation
- ○ elimina qualquer possibilidade de qualquer tipo de problema adicional
- ○ nenhum dos mencionados

Se a thread A está esperando por um recurso compartilhado, exclusivo e não preemptivo que é mantido pela thread B e a thread B está esperando por um recurso exatamente com as mesmas características que é mantido por A, isso é chamado: *

- ○ starvation
- ◉ deadlock
- ○ espera ocupada
- ○ exclusão mútua
- ○ sincronização

Para garantir que não surjam erros no problema dos leitores-escritores, os _____ recebem acesso exclusivo ao objeto compartilhado. *

- ○ leitores
- ◉ escritores
- ○ leitores e escritores
- ○ nenhum dos mencionados

O código do monitor abaixo é suficiente para implementar o problema dos leitores-escritores em Java. *

```
class MonitorRW {
  protected int data = 0;  // the "database"
  public synchronized void read(String tid) {
    System.out.println(tid + " read:   " + data);
  }
  public synchronized void write(String tid) {
    data++;
    System.out.println(tid + " wrote:  " + data);
  }
}
```

- ○ Verdadeiro
- ◉ Falso

Barreira é uma construção de sincronização onde um conjunto de processos (ou threads) sincroniza globalmente, ou seja, cada processo no conjunto chega à barreira e espera que todos os outros cheguem e, em seguida, todos os processos deixam a barreira. Considere que o número de processos num determinado conjunto seja 3 e que S é um semáforo binário com as funções P e V usuais. Considere a seguinte implementação em C de uma barreira. As variáveis process_arrived e process_left são compartilhadas entre todos os processos e são inicializadas com zero. Em um programa concorrente, todos os 3 processos chamam a função de barreira quando precisam sincronizar globalmente. A implementação da barreira abaixo está incorreta. Qual das seguintes opções é verdadeira? *

```
void barrier (void) {
1:  P(S);
2:  process_arrived++;
3.  V(S);
4:  while (process_arrived !=3);
5:  P(S);
6:  process_left++;
7:  if (process_left==3) {
8:    process_arrived = 0;
9:    process_left = 0;
10: }
11: V(S);
}
```

- ○ A implementação da barreira está errada devido ao uso do semáforo binário S.
- ◉ A implementação da barreira pode levar ao deadlock se os processos invocarem duas vezes a função barrier() uma após a outra no código.
- ○ As linhas de 6 a 10 não precisam estar dentro de uma seção crítica.
- ○ A implementação da barreira está correta se houver apenas dois processos em vez de três.

Qual das diretivas abaixo força as threads a esperar até que todas cheguem a este mesmo ponto? *

- ( ) #pragma omp atomic
- (•) #pragma omp barrier
- ( ) #pragma omp critical
- ( ) #pragma omp sections