Lista de Exercícios 3

1) Em uma aplicação concorrente que controla saldo bancário em contas correntes, dois processos compartilham uma região de memória onde estão armazenados os saldos dos clientes A e B. Os processos executam, concorrentemente os seguintes passos:

Processo 1 (Cliente A)

```
/* saque em A */
1a. x = saldo_do_cliente_A;
1b. x = x - 200;
1c. saldo_do_cliente_A = x;

/* deposito em B */
1d. x = saldo_do_cliente_B;
1e. x = x + 100;
1f. saldo_do_cliente_B = x;
```

Processo 2 (Cliente B)

```
/*saque em A */
2a. y = saldo_do_cliente_A;
2b. y = y - 100;
2c. saldo_do_cliente_A = y;

/* deposito em B */
2d. y = saldo_do_cliente_B;
2e. y = y + 200;
2f. saldo_do_cliente_B = y;
```

Supondo que os valores dos saldos de A e B sejam, respectivamente, 500 e 900, antes de os processos executarem, pede-se?

a) Quais os valores corretos esperados para os saldos dos clientes A e B após o término da execução dos processos?

b) Quais os valores finais dos saldos dos clientes se a sequência temporal de execução das operações for: 1a, 2a, 1b, 2b, 1c, 2c, 1d, 2d, 1e, 2e, 1f, 2f?

c) Utilizando semáforos, proponha uma solução que garanta a integridade dos saldos e permita o maior compartilhamento possível dos recursos entre os processos.

2) Diferencie os tempos de processador, espera, turnaround e resposta.

3) Existem quatro processos (P1, P2, P3 e P4) na fila de pronto, com tempos de UCP estimados em 9, 6, 3 e 5 Uts, respectivamente. Em que ordem os processos devem ser executados para minimizar o tempo de turnaround dos processos?

4) O que é o contexto de hardware de um processo e como é a implementação da troca de contexto?

5) Defina o que é uma aplicação concorrente e dê um exemplo de sua utilização.

6) Considere uma aplicação que utilize uma matriz na memória principal para a comunicação entre vários processos concorrentes. Que tipo de problema pode ocorrer quando dois ou mais processos acessam uma mesma posição da matriz?

7) O que é exclusão mútua e como é implementada?

8) Seja o Algoritmo de Dekker – 2a abordagem – mostrado abaixo:

int flag[2]; /*variável global*/

Processo 0	Processo 1
.	.
while flag[1] do { };	while flag[0] do { };
/***/	/***/
flag[0] = TRUE;	flag[1] = TRUE;
<RC>	<RC>
flag[0] = FALSE;	flag[1] = FALSE;
.	.

Há algum problema de execução? Explique.

9) Seja o Algoritmo de Dekker – 3a abordagem – mostrado abaixo:

int flag[2]; /*variável global*/

Processo 0	Processo 1
.	.
flag[0] = TRUE;	flag[1] = TRUE;
/***/	/***/
while flag[1] do { };	while flag[0] do { };
<RC>	<RC>
flag[0] = FALSE;	flag[1] = FALSE;
.	.

Há algum problema de execução? Explique.

10) Seja o Algoritmo de Dekker – 4a abordagem – mostrado abaixo:

int flag[2]; /*variável global*/

Processo 0	Processo 1
flag[0] = TRUE;	flag[1] = TRUE;
/***/	/***/
while flag[1] do	while flag[0] do
{ flag[0] = FALSE;	{flag[1] = FALSE;
<delay randômico>	<delay randômico>
flag[0] = TRUE;}	flag[1] = TRUE;}
<RC>	<RC>
flag[0] = FALSE;	flag[1] = FALSE;
.	.

Há algum problema de execução? Explique.