Gabarito Prova 2

1)

Questão 1

a) Necessários

	A	B	C	D
P_1	0	0	0	0
P_2	0	7	5	0
P_3	1	0	0	2
P_4	0	0	2	0
P_5	0	6	4	2

b) Disponíveis [1 5 2 0]

Passo 1: Executa P_1 → [1532]

Passo 2: Executa P_3 → [2886]

Passo 3: Executa P_4 → [2 14 11 8]

Passo 4: Executa P_5 → [2 14 12 12]

Passo 5: Executa P_2 → [3 14 12 12]

Está no estado seguro pois consegue finalizar todos os processos respeitando a ordem de execução.

c) O processo 1 pode requerer 1 recurso C e 2 recursos D. De acordo com o estado atual do sistema (snapshot) essas requisições já foram realizadas uma vez que a matriz alocada tem 1 recurso C e 2 recursos D. Portanto, essas requisições não pode ser atendidas.

2)

Questão 02

a) while (true){

```
while (cont != 0);
    região crítica();
    cont = -1;
    região não crítica ();
}
```

b) Sim, esse problema pode ocorrer com a solução baseada em alternância obrigatória. Caso o processo 2 entre na região não crítica o valor da variável cont será "-1". Caso ocorra uma interrupção pelo escalonador de processos para executar o processo 1. Esse poderá entrar na região crítica. Quando terminar mudará o valor de cont para o valor "0". Assim poderá entrar na região não crítica. Se continuar executando e terminar a sua região não crítica não poderá mais entrar na região crítica, pois o valor de cont = 0 e essa condição impede de entrar na região crítica. Com isso, processo 2 impede processo 1 de entrar na região crítica.

3)

Coffman et al. (1971) mostraram que quatro condições devem ser atendidas para que ocorrer um deadlock:

1) Condição de exclusão mútua. Cada recurso está atualmente atribuído a exatamente um processo ou está disponível. Se houver uma cenário em que a condição de controle de sincronização não é respeitada irá ocorrer um deadlock, como por exemplo, a variável de controle count.
2) Condição de uso e espera. Os processos que atualmente mantêm recursos que foram concedidos anteriormente podem solicitar novos recursos. Se houver um cenário em que um processo tem controle sobre um recurso mas é retirado da execução, por exemplo, uso do algoritmo de prioridade, o processo escalonado de mais alta prioridade pode requerer o recurso já alocado ao processo anterior e não irá concluir.
3) Condição de não preempção. Os recursos concedidos anteriormente não podem ser retirados à força de um processo. Eles devem ser explicitamente liberados pelo processo que os está mantendo. O uso de um recurso não preemptivo como por exemplo, o uso de uma unidade de DVD.
4) Condição de espera circular. Deve haver uma lista circular de dois ou mais processos processos, cada um dos quais está aguardando um recurso mantido pelo próximo membro. Se houver um cenário em processos obtem recursos mas depende de recursos de outros processos isso poderá gerar um loop entre os processos e recursos.

4) Quando os threads são usados em um programa ocorrerá uma concorrência por recursos compartilhados quando o processo estiver ativo. Como a variável i é um recurso compartilhando e não há nenhum tipo de controle para o acesso a informação, duas ou mais threads poderão acessar a mesma informação. Então, a mensagem impressa como resultado poderá ocorrer uma vez que há uma disputa de recurso compartilhado (variável i) sem controle de sincronização.

5) V - V - F - V - V - F