EXERCÍCIO 4.1

Considere o seguinte algoritmo, executado concorrentemente, numa máquina com um só núcleo de execução, pelos processos P0 e P1, sendo **`lock`** uma variável inteira (de valor inicial 0) partilhada por P0 e P1. Assuma como não-atómicas (divisíveis) as secções de código de entrada e saída.

```
while (1) {
            while (lock == 1) {;}   // A        // secção de entrada
            lock = 1;               // B

            ...                     // C        // secção crítica

            lock = 0;               // D        // secção de saída

            ...                     // E        // secção restante
}
```

a) Com base na tabela abaixo, apresente uma situação que demonstre que este algoritmo não respeita o requisito da **Exclusão Mútua** de uma solução para o Problema da Secção Crítica com 2 processos. Para cada instante **t+n** (n=0,1,...), preencha cada célula das colunas **P0** e **P1** com uma letra correspondente a uma ação do programa principal (A, B, C ou D), ou deixe-a vazia caso o processo em causa não realize nenhuma ação naquele instante (por cada linha, só pode haver uma ação realizada). Preencha ainda a coluna **lock** com o valor que esta variável assume logo após a execução da ação realizada em cada linha. Nota: a linha do instante **t+0** já está preenchida, e a tabela tem o número mínimo de linhas necessárias para a demonstração; no entanto, caso encontre conveniente, pode acrescentar as linhas que entender necessárias para suportar a sua solução.

Instante	P0	P1	lock
t+0	**A**		**0**
t+1		**A**	**0**
t+2		**B**	**1**
t+3	**B**		**1**
t+4	**C**		**1**
t+5		**C**	**1**

b) Usando a mesma metodologia, demonstre que a **Espera Limitada** também não é respeitada (assuma que a 1ª linha da tabela está preenchida da mesma forma que em a), e que poderá acrescentar mais linhas para suportar a sua solução).

Instante	P0	P1	lock
t+0	**A**		**0**
t+1	**B**		**1**
t+2	**C**		**1**
t+3		**A**	**1**
t+4	**D**		**0**
t+5	**E**		**0**
t+6	**A**		**0**
t+7	**B**		**1**
t+8	**C**		**1**

Notas sobre os diagramas temporais (tabelas) usados neste exercício:
- um diagrama temporal é uma representação gráfica ou tabela que mostra a sequência de instruções executadas pelos processos envolvidos, incluindo a sua alternância na ocupação da CPU;
- nestes diagramas assumem-se várias simplificações: i) qualquer ação de um processo (A,B,C,D) consome o mesmo tempo; ii) a comutação de processos ocorre de forma instantânea; iii) não há mais processos a competir pela CPU além de P0 e P1;
- esta metodologia pode ser aplicada a qualquer proposta de solução do Problema da Secção Crítica (algoritmos da Solução de Peterson, algoritmos baseados nas instruções-máquina TestAndSet ou CompareAndSwap, Trincos e Semáforos).

EXERCÍCIO 4.2

Dois processos, A e B, executam concorrentemente uma sequência de 6 ações. Algumas ações já estão definidas, bem como a sua ordem: no processo A, a 2ª e 3ª ações são a invocação das funções f1() e f2(); no processo B, a 3ª e a 4ª ações são a invocação das funções g1() e g2(). As outras ações estão por definir, mas têm de ser operações Wait e Signal sobre semáforos com espera passiva, sendo que para um semáforo S a operação Wait(S) deve sempre preceder a operação Signal(S), podendo haver lugar a outras ações entre essas duas operações.

Processo A	Processo B
f1();	
f2();	g1();
	g2();

a) Assumindo que os processos A e B partilham 2 semáforos X e Y, de valor inicial 1, e que cada processo tem de realizar operações Wait e Signal sobre ambos os semáforos, complete a tabela abaixo com as ações de cada processo ao longo do tempo, de forma a garantir que todas as ações previstas são realizadas (ou seja, não há *deadlock*). Assuma ainda a execução numa máquina com um só núcleo de execução, donde em cada instante só pode ser executada uma ação (ou em A, ou em B). Preencha ainda as duas últimas colunas com o valor do contador dos semáforos logo após a execução de cada ação. Nota: a linha do instante **t+0** já está preenchida, e a tabela tem o número exato de linhas necessárias para a demonstração solicitada.

Instante	Processo A	Processo B	Contador de X	Contador de Y
t+0	Wait(X)		0	1
t+1	**f1();**		**0**	**1**
t+2	**f2();**		**0**	**1**
t+3	**Signal(X)**		**1**	**1**
t+4	**Wait(Y)**		**1**	**0**
t+5	**Signal(Y)**		**1**	**1**
t+6		**Wait(X)**	**0**	**1**
t+7		**Wait(Y)**	**0**	**0**
t+8		**g1();**	**0**	**0**
t+9		**g2();**	**0**	**0**
t+10		**Signal(Y);**	**0**	**1**
t+11		**Signal(X)**	**1**	**1**

b) Usando a mesma metodologia, preencha a tabela abaixo de forma a gerar um cenário de *deadlock* (assuma que a 1ª linha da tabela está preenchida da mesma forma que em a) e que, desta vez, pode não necessitar de preencher todas as linhas da tabela).

Instante	Processo A	Processo B	Contador de X	Contador de Y
t+0	Wait(X)		0	1
t+1		**Wait(Y)**	**0**	**0**
t+2		**Wait(X)**	**-1**	**0**
t+3	**f1();**		**-1**	**0**
t+4	**f2();**		**-1**	**0**
t+5	**Wait(Y)**		**-1**	**-1**
t+6				
t+7				
t+8				
t+9				
t+10				
t+11				

Notas sobre os diagramas temporais (tabelas) usados neste exercício:
- um diagrama temporal é uma representação gráfica ou tabela que mostra a sequência de instruções executadas pelos processos envolvidos, incluindo a sua alternância na ocupação da CPU;
- nestes diagramas assumem-se várias simplificações: i) qualquer ação de um processo (A ou B) consome o mesmo tempo; ii) a comutação de processos ocorre de forma instantânea; iii) não há mais processos a competir pela CPU além de A e B.