

Nome do aluno: \_\_\_\_\_ Matrícula do aluno: \_\_\_\_\_

**1)(1,6 ponto)** A Figura 1 apresenta um *schedule* que é **equivalente em conflito** a pelo menos **um** *schedule* serial. Com base nesse *schedule*, responda as seguintes questões.

- a) (0,8 ponto) Mostre **todos** *schedules* seriais que são equivalentes a esse *schedule*.
- b) (0,8 ponto) Mudando a **última** instrução de **T3** para a **leitura de outro item** é possível gerar um *schedule* que **não** seja equivalente a um *schedule* serial. Que mudança seria essa? Descreva o grafo de precedência que seria gerado.

T1	T2	T3	T4	T5
		Read(A)		
	Write(F)			
				read(F)
Read(B)				
Read(C)				
		Read(C)		
		Read(C)		
			Read(F)	
	Write(B)			
				Write(C)
Write(A)				
Write(D)				

Figura 1 – Schedule serial sem conflito

**2) (1,5 ponto)** A Figura 2 apresenta um *schedule* parcial ainda sem instruções de *commit/abort*. Complete o *schedule* com instruções de *commit/abort* de modo que ele se torne **irrecuperável**. Explique **porque** ele é irrecuperável.

T6	T7	T8
Read(B)		
	Read(A)	
		Read(A)
Write(C)		
	write(B)	
		Read(B)

Figura 2 – Instruções de três transações (T6, T7 e T8)

**Observação:** Para responder as questões três e quatro é necessário simular como o gerenciador de *locks* controla a aquisição e liberação de acesso aos recursos requisitados pelas transações da Figura 3. Na simulação, considere os seguintes aspectos:

- As transações são atendidas pelo gerenciador em ciclos de execuções, onde em cada ciclo todas transações tem o direito de tentar executar uma instrução.
- Dentro de cada ciclo a ordem de execução é determinada pelo número da transação.
- O gerenciador utiliza o protocolo *two phase locking*.

**3)(2 pontos)** Monte um *schedule* de execução usando a estratégia de prevenção de *deadlock* “*wait-die*”. As esperas **não** precisam ser indicadas. Os *aborts* e *commits* sim.

**4)(2 pontos)** Para este exercício, considere que nenhuma estratégia de prevenção de *deadlock* seja usada. Ainda, considere que todos os itens de dados tenham 0 (zero) como valor inicial. Com base nisso, responda as seguintes questões:

- a) Mostre o log de recuperação que seria gerado para o esquema de **modificação postergada**.
- b) Caso ocorresse um erro **imediatamente antes** do commit de T2, quais ações de log teriam que ser realizadas?

T9: read(A); Write(B=1). Read(G)	T10: read(B); Write(C=2).	T11: read(C); Write(D=3). Write(G=9)	T12: read(D); Write(F=4).	T13: read(E); read(F) write(C=5)
---	---------------------------------	---	---------------------------------	---

Figura 3 – Instruções de cinco transações (T9, T10, T11, T12 e T13)

**5)(1 ponto)** No nível de isolamento *ReadCommitted* a transação libera um *lock* compartilhado assim que acaba de realizar a leitura desejada. Esse comportamento pode levar a um problema de consistência. Descreva esse problema.

**6)(1 ponto)** Uma característica desejável é que o gerenciador de transações gere *schedules* livres de *rollback* em cascata. Contudo, é imprescindível que ele gere *schedules* recuperáveis. Explique porque um deles é desejável e o outro é imprescindível.

**7)(1 ponto)** É possível evitar que *deadlocks* ocorram usando estratégias baseadas em *timestamp*, como *wound-wait* e *wait-die*. No entanto, na prática é comum deixar que ocorra um *deadlock* antes de remedíá-lo. Como o gerenciador de transações detecta um *deadlock*? O que ele faz para que o *deadlock* seja eliminado?