

Instruções gerais: O BOCA é um sistema de correção automática de exercícios que verifica se o resultado gerado pelo seu programa satisfaz casos de teste pré-definidos. Portanto, é necessário seguir estritamente os formatos especificados na questão. Lembre-se que os exemplos dados servem para facilitar o entendimento e podem não cobrir todos os casos de teste que serão usados.

Diz-se que uma transação segue o **protocolo de bloqueio em duas fases** (também conhecido como **2PL básico**) se todas as operações de bloqueio (`read_lock`, `write_lock`) precedem a primeira operação de desbloqueio na transação. Uma variação mais restritiva do 2PL é o **2PL rigoroso**, que garante *schedules* estritos, em que as transações não podem ler nem gravar um item X até que a última transação que gravou X tenha sido confirmada (ou cancelada). Nessa variação, uma transação T não libera nenhum de seus bloqueios (exclusivo ou compartilhado) até depois de confirmar ou abortar. Logo, nenhuma outra transação pode ler ou gravar um item que é gravado por T a menos que T tenha sido confirmado, levando a um *schedule* estrito para facilidade de recuperação. O 2PL rigoroso não é livre de *deadlock*.

Escreva uma consulta que verifica se as transações seguem ou não o protocolo descrito.

Entrada:

Considere a existência da tabela **Schedule**, na qual cada linha representa a chegada de uma operação pertencente a uma dada transação (o número de transações presentes no *schedule* pode variar). A tabela possui 4 colunas: a primeira representa o tempo de chegada (`time`), a segunda o identificador da transação (`#t`), a terceira a operação (`read_lock` : bloqueio (compartilhado) para leitura de um item, `write_lock` : bloqueio (exclusivo) para escrita/gravação de um item, `read_item` : desbloqueio de um item, `read_item` : leitura de um item, `write_item` : escrita de um item, `commit` : confirmação ou `abort` : aborto/*rollback*) e a quarta o item de dados (atributo) que será bloqueado/desbloqueado/lido/escrito (quando aplicável). As linhas da tabela estão ordenadas logicamente pelo valor na primeira coluna, que indica o carimbo (rótulo) de tempo (*timestamp*) de chegada (quanto menor o valor, mais antiga a operação).

Saída:

A saída deve ser uma tabela contendo uma coluna chamada `RESP` com o valor 1, se o *schedule* seguir o protocolo descrito; caso contrário, 0.

Exemplo 01

time	#t	op	attr
1	1	read_lock	Y
2	1	read_item	Y
3	1	unlock	Y
4	1	write_lock	X
5	1	read_item	X
6	1	write_item	X
7	1	unlock	X
8	1	commit	-

Saída

0

Exemplo 01

time	#t	op	attr
21	2	read_lock	Y
42	2	read_item	Y
43	2	write_lock	X
54	2	read_item	X
65	2	write_item	X
76	2	abort	-
77	2	unlock	X
89	2	unlock	Y

Saída

1