1 ACID Properties (8 Points)

The following example shows three interleaved executed transactions. The first transaction transfers 50 euros from account A to account B, while the second transaction transfers 10 euros from account B to account C, and the third transaction transfers 70 euros from account C to account A.

	T_1	T_2	T_3
1			$bal_{a3} = \text{r(balA)}$
2		$bal_{c2} = \text{r(balC)}$	
3	$bal_{a1} = \text{r(balA)}$		
4		$bal_{c2} = bal_{c2} + 10$	
5	$bal_{a1} = bal_{a1} - 50$		
6			$bal_{a3} = bal_{a3} + 70$
7		$\text{w(balC=}bal_{c2}\text{)}$	
8			$\text{w(balA=}bal_{a3}\text{)}$
9			$bal_{c3} = \text{r(balC)}$
10	$\text{w(balA=}bal_{a1}\text{)}$		
11			$bal_{c3} = bal_{c3} - 70$
12			$\text{w(balC=}bal_{c3}\text{)}$
13			commit
	The current state of all transactions is persistently written to disk.		
14		$bal_{b2} = \text{r(balB)}$	
15	$bal_{b1} = \text{r(balB)}$		
16		$bal_{b2} = bal_{b2} - 10$	
17		$\text{w(balB=}bal_{b2}\text{)}$	
18		commit	
19	$bal_{b1} = bal_{b1} + 50$		
20	$\text{w(balB=}bal_{b1}\text{)}$		
21	commit		

Assume that all write operations are performed in main memory and thus become immediately visible to all transactions. However, changes in main memory are not persistent, which is only the case for changes of data on disk.

Also assume that our consistency condition is that the sum of all account balances must not be changed, otherwise money can be lost or generated.

For each of the four ACID properties, explain how and where (responsible lines or operations) they are potentially violated in the example above.

2 Serialisability Theory (5 Points)

Consider the following transactions.

T_1	T_2	T_3
read(A)	read(B)	read(C)
A := A + 15	B := B − 71	C := C + 40
write(A)	write(B)	write(C)
	read(C)	read(A)
	C := C + 71	A := A ∗ 3
	write(C)	write(A)

1. Identify all pairs of conflicting operations. (1 Point)

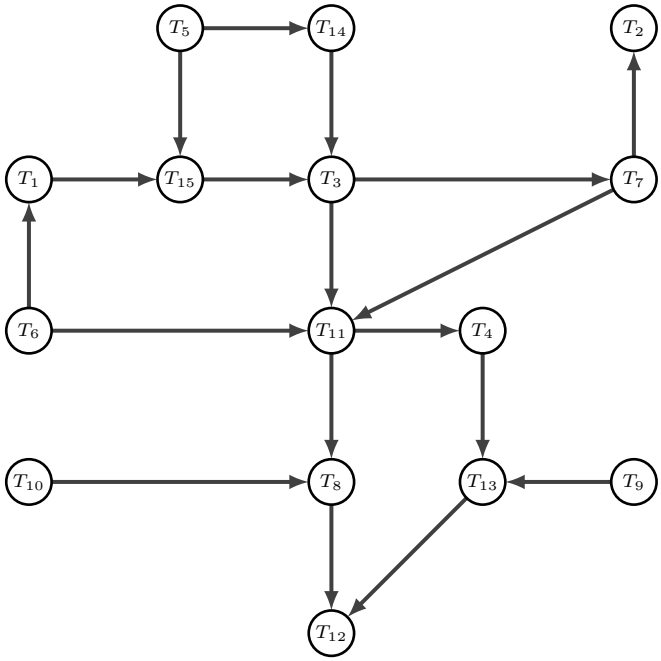
2. The following four schedules are given:

	Schedules
S_1	$r_1(A) \rightarrow r_2(B) \rightarrow r_3(C) \rightarrow w_3(C) \rightarrow w_1(A) \rightarrow r_3(A) \rightarrow w_2(B) \rightarrow r_2(C) \rightarrow w_3(A) \rightarrow w_2(C)$
S_2	$r_2(B) \rightarrow r_3(C) \rightarrow w_2(B) \rightarrow r_2(C) \rightarrow r_1(A) \rightarrow w_2(C) \rightarrow w_3(C) \rightarrow r_3(A) \rightarrow w_3(A) \rightarrow w_1(A)$
S_3	$r_1(A) \rightarrow r_3(C) \rightarrow w_3(C) \rightarrow r_2(B) \rightarrow w_2(B) \rightarrow r_3(A) \rightarrow r_2(C) \rightarrow w_1(A) \rightarrow w_2(C) \rightarrow w_3(A)$
S_4	$r_3(C) \rightarrow r_2(B) \rightarrow w_3(C) \rightarrow r_3(A) \rightarrow w_2(B) \rightarrow r_2(C) \rightarrow w_3(A) \rightarrow r_1(A) \rightarrow w_2(C) \rightarrow w_1(A)$

   For each schedule, create the conflict graph and argue whether the schedule is conflict serialisable. If the schedule is conflict serialisable, reorder the non-conflict operations such that you obtain a serial, conflict equivalent schedule and state it. (4 Points)

3 Conflict Graphs (4 Points)

Consider the following conflict graph.



1. Provide a serial schedule of the transactions in the conflict graph such that all conflicts are resolved. (1 Point)

2. Assume that you can run transactions in parallel and all transactions take one unit of time to complete their operations. Provide a schedule **for 2 threads and for 4 threads** that uses parallel execution to execute the transactions in the shortest possible time. Use the following table to represent a schedule, with the rows corresponding to the threads and the columns corresponding to the units of time. Further, what is the maximum number of threads that is reasonable for the given conflict graph? Justify your answer. (3 Points)

	1	2	3	...
Thread 1				
Thread 2				
⋮				

4 Cycles in Conflict Graphs (3 Points)

In the lecture you have seen how you can identify whether a given schedule is conflict serialisable by examining the conflict graph regarding cycles.
In the attached notebook you can find the class `ConflictGraph`. It contains a dictionary mapping from vertices $v \in V$ to a list of vertices U_v, where for each $u \in U_v$ it holds that (v, u) is a directed edge in the graph.

Your task is to implement the function `contains_cycle` that analyses whether the corresponding graph contains a cycle. In case the conflict graph contains a cycle, your function should return `True` and `False` otherwise. Your solution should have a runtime of $\mathcal{O}(n + m)$, with n being the number of vertices and m the number of edges in the conflict graph.

Note: You are not allowed to use any additional imports!

Submission

Solutions must be submitted in teams of 3 to 4 students by July, 11 2024, 10:00 a.m. via your personal status page in CMS using the Team Groupings functionality. Late submissions will not be graded!

Please note that there are two submissions, under **Theoratical** you submit your solutions to exercises 1, 2 and 3 as a PDF file.

Your solution to exercise 4 must be handed in under **Practical** as txt file.

Make sure that you only copy the complete Jupyter cells you want to add and that indentation and formatting are correct.