Problem Set 4, Part I

Problem 1: Timestamp-based concurrency control

1.1

request	**response of the system**	**any changes to state for A *and/or* explanation of why action wasn't accepted**
r2(A)	allowed	RTS = 20
w4(A)	allowed	WTS = 40
r3(A)	denied; roll back	TS(T3) < WTS(A)
w1(A)	denied; roll back	TS(T1) < RTS(A)
w2(A)	Ignored	TS(T2) >= RTS(A) and TS(T2) < WTS(A)
r5(A)	allowed	RTS = 50

1.2

request	**response of the system**	**any changes to state for A *and/or* explanation of why action wasn't accepted**
r2(A)	allowed	RTS = 20
w4(A)	allowed	WTS = 40, c = false
r3(A)	denied; roll back	TS(T3) < WTS(A)
w1(A)	denied; roll back	TS(T1) < RTS(A)
w2(A)	denied; make wait	TS(T2) >= RTS(A) and TS(T2) < WTS(A), but c = false
r5(A)	denied; make wait	TS(T5) >= WTS(A), but c = false
c4	allowed	c = true
w2(A)	ignored	No changes
r5(A)	allowed	RTS = 50
c2	allowed	No changes
c5	allowed	No changes

1.3

request	**response of the system**	**any changes to state for A *and/or* explanation of why action wasn't accepted**
r2(A)	allowed to read A(0)	RTS(A(0)) = 20
w4(A)	allowed to write A	create A(40) with RTS = 0
r3(A)	allowed to read A(0)	RTS(A(0)) = 30
w1(A)	denied; rollback	TS(T1) < RTS(A(0))
w2(A)	denied; rollback	TS(T2) < RTS(A(0))
r5(A)	Allowed to read A(40)	RTS(A(40)) = 50

Problem 2: Replication and distributed concurrency control

2.1) 11 copies of each item, fully distributed locking

voting scheme	***would it work? (yes/no)***	***explanation***
1a) update 6, read 5	no	• s = n - x (5 = 11 - 6) • If one transaction has a global shared lock, someone else can get a global exclusive lock • If one transaction has a global exclusive lock, someone else can get a global shared lock
1b) update 5, read 8	no	• w < n/2 (5 < 11/2) • Two transactions can both get a global exclusive lock at the same time.
1c) update 8, read 4	yes	• w > n/2 (8 > 11/2) and s > n - x (4 > 11 - 8) • Two transactions can't both get a global exclusive lock at the same time • If one txn has a global shared lock, no one else can get a global exclusive lock • If one transaction has a global exclusive lock, no one else can get a global shared lock
1d) update 3, read 10	no	• w < n/2 (3 < 11/2) • Two transactions can both get a global exclusive lock at the same time.

2.2) 11 copies of each item, primary-copy locking

voting scheme	***would it work? (yes/no)***	***explanation***
2a) update 6, read 5	no	• r = n - w (5 = 11 - 6) • If there are 11 copies and 6 are updated, that means there are 5 unupdated. This means that it is possible for there to be 5 reads of all 5 updated copies.
2b) update 5, read 8	yes	• r > n - w (8 > 11 - 5) • If there are 11 copies and 5 are updated, that means there are 6 unupdated. This means that out of the 8 reads of copies, at least one has to be updated.
2c) update 8, read 4	yes	• r > n - w (4 > 11 - 8) • If there are 11 copies and 8 are updated, that means there are 3 unupdated. This means that out of 4 reads of copies, at least one has to be updated.

2d) update 3, read 10	yes	● r > n - w (10 > 1 1- 3) ● If there are 11 copies and 3 are updated, that means there are 8 unupdated. This means that out of 10 reads of copies, at least one has to be updated.

2.3)

Configuration 2c would be a good choice. It only requires that 4 copies are read. It also requires that only the primary copy be locked, so it will likely be more efficient than other options for a workload with a lot of reads. The main disadvantage is that if the primary copy of a given item becomes unavailable, it becomes impossible to read or write that item.

2.4)

Configuration 2d would be a good choice. It only requires that 3 copies are updated. It also only requires that the primary copy be locked, so it will likely be more efficient than the other options for a workload that has a lot of updates.The main disadvantage is that if the primary copy of a given item becomes unavailable, it becomes impossible to read or write that item.