

Ronny Bull  
CS541  
Ch22 Problem 22.25

### A) Timestamp Ordering Algorithm applied to Schedule E

Initial TS Values:

read\_TS(X) = 0  
read\_TS(Y) = 0  
read\_TS(Z) = 0  
write\_TS(X) = 0  
write\_TS(Y) = 0  
write\_TS(Z) = 0

Transaction TS Values (based on the time the first transaction starts in the group):

TS(T1) = 6  
TS(T2) = 1  
TS(T3) = 4

Begin Processing:

T2: read\_item(Z)  
write\_TS(Z) < TS(T2) //check is good proceed  
execute read\_item(Z)  
set read\_TS(Z) ← max(read\_TS(Z),TS(T2)) = 1 //setting to max of read\_TS(Z)=0 or TS(T2)=1  
//read\_TS(Z) is now 1

T2: read\_item(Y)  
write\_TS(Y) < TS(T2) //check is good proceed  
execute read\_item(Y)  
set read\_TS(Y) ← max(read\_TS(Y),TS(T2)) = 1 //setting to max of read\_TS(Y)=0 or TS(T2)=1  
//read\_TS(Y) is now 1

T2: write\_item(Y)  
write\_TS(Y) < TS(T2) && read\_TS = TS(T2) //check is good proceed  
execute write\_item(Y)  
set write\_TS(Y) ← max(write\_TS(Y),TS(T2)) = 1 //setting to max of write\_TS(Y)=0 or TS(T2)=1  
//write\_TS(Y) is now 1

T3: read\_item(Y)  
write\_TS(Y) < TS(T3) //check is good proceed  
execute read\_item(Y)  
set read\_TS(Y) ← max(read\_TS(Y),TS(T3)) = 4 //setting to max of read\_TS(Y)=0 or TS(T3)=4  
//read\_TS(Y) is now 4

T3: read\_item(Z)  
write\_TS(Z) < TS(T3) //check is good proceed  
execute read\_item(Z)  
set read\_TS(Z) ← max(read\_TS(Z),TS(T3)) = 4 //setting to max of read\_TS(Z)=1 or TS(T3)=4  
//read\_TS(Z) is now 4

T1: read\_item(X)  
write\_TS(X) < TS(T1) //check is good proceed  
execute read\_item(X)  
set read\_TS(X) ← max(read\_TS(X),TS(T1)) = 6 //setting to max of read\_TS(X)=0 or TS(T1)=6  
//read\_TS(X) is now 6

T1: write\_item(X)  
read\_TS(X) = TS(T1) && write\_TS(X) < TS(T1) //check is good proceed  
execute write\_item(X)  
set write\_TS(X) ← max(write\_TS(X),TS(T1)) = 6 //setting to max of write\_TS(X)=0 or TS(T1)=6  
//write\_TS(X) is now 6

T3: write\_item(Y)  
read\_TS(Y) = TS(T3) && write\_TS(Y) < TS(T3) //check is good proceed  
execute write\_item(Y)  
set write\_TS(Y) ← max(write\_TS(Y),TS(T3)) = 4 //setting to max of write\_TS(Y)=0 or TS(T3)=4  
//write\_TS(Y) is now 4

T3: write\_item(Z)  
read\_TS(Z) < TS(T3) && write\_TS(Z) < TS(T3) //check is good proceed  
execute write\_item(Z)  
write\_TS(Z) ← max(write\_TS(Z),TS(T3)) = 4 //setting to max of write\_TS(Z)=0 or TS(T3)=4  
//write\_TS(Z) is now 4

T2: read\_item(X)  
write\_TS(X) > TS(T2) //check failed abort & rollback

Result: Schedule E will not be able to complete since the write\_TS(X) value is greater than the TS(T2) value which causes the check to fail and the transaction to be rolled back.

## B) Timestamp Ordering Algorithm applied to Schedule F

Initial TS Values:

read\_TS(X) = 0  
read\_TS(Y) = 0  
read\_TS(Z) = 0  
write\_TS(X) = 0  
write\_TS(Y) = 0  
write\_TS(Z) = 0

Transaction TS Values (based on the time the first transaction starts in the group):

TS(T1) = 3  
TS(T2) = 7  
TS(T3) = 1

Begin Processing:

T3: read\_item(Y)  
write\_TS(Y) < TS(T3) //check is good proceed  
execute read\_item(Y)  
set read\_TS(Y) ← max(read\_TS(Y),TS(T3)) = 1 //setting to max of read\_TS(Y)=0 or TS(T3)=1  
//read\_TS(Y) is now 1

T3: read\_item(Z)  
write\_TS(Z) < TS(T3) //check is good proceed  
execute read\_item(Z)  
set read\_TS(Z) ← max(read\_TS(Z),TS(T3)) = 1 //setting to max of read\_TS(Z)=0 or TS(T3)=1  
//read\_TS(Z) is now 1

T1: read\_item(X)  
write\_TS(X) < TS(T1) //check is good proceed  
execute read\_item(X)  
set read\_TS(X) ← max(read\_TS(X),TS(T1)) = 3 //setting to max of read\_TS(X)=0 or TS(T1)=3  
//read\_TS(X) is now 3

T1: write\_item(X)  
read\_TS(X) = TS(T1) && write\_TS(X) < TS(T1) //check is good proceed  
execute write\_item(X)  
set write\_TS(X) ← max(write\_TS(X),TS(T1)) = 3 //setting to max of write\_TS(X)=0 or TS(T1)=3  
//write\_TS(X) is now 3

T3: write\_item(Y)  
read\_TS(Y) = TS(T3) && write\_TS(Y) < TS(T3) //check is good proceed  
execute write\_item(Y)  
set write\_TS(Y) ← max(write\_TS(Y),TS(T3)) = 1 //setting to max of write\_TS(Y)=0 or TS(T3)=1  
//write\_TS(Y) is now 1

T3: write\_item(Z)  
read\_TS(Z) = TS(T3) && write\_TS(Z) < TS(T3) //check is good proceed  
execute write\_item(Z)  
set write\_TS(Z) ← max(write\_TS(Z),TS(T3)) = 1 //setting to max of write\_TS(Z)=0 or TS(T3)=1  
//write\_TS(Z) is now 1

T2: read\_item(Z)  
write\_TS(Z) < TS(T2) //check is good proceed  
execute read\_item(Z)  
set read\_TS(Z) ← max(read\_TS(Z),TS(T2)) = 7 //setting to max of READ\_TS(Z)=1 or TS(T2)=7  
//read\_TS(Z) is now 7

T1: read\_item(Y)  
write\_TS(Y) < TS(T1) //check is good proceed  
execute read\_item(Y)  
set read\_TS(Y) ← max(read\_TS(Y),TS(T1)) = 3 //setting to max of read\_TS(Y)=1 or TS(T1)=3  
//read\_TS(Y) is now 3

T1: write\_item(Y)  
read\_TS(Y) = TS(T1) && write\_TS(Y) < TS(T1) //check is good proceed  
execute write\_item(Y)  
set write\_TS(Y) ← max(write\_TS(Y),TS(T1)) = 3 //setting to max of write\_TS(Y)=1 or TS(T1)=3  
//write\_TS(Y) is now 3

T2: read\_item(Y)  
write\_TS(Y) < TS(T2) //check is good proceed  
execute read\_item(Y)  
set read\_TS(Y) ← max(read\_TS(Y),TS(T2)) = 7 //setting to max of read\_TS(Y)=3 or TS(T2)=7  
//read\_TS(Y) is now 7

T2: write\_item(Y)  
read\_TS(Y) = TS(T2) && write\_TS(Y) < TS(T2) //check is good proceed  
execute write\_item(Y)  
set write\_TS(Y) ← max(write\_TS(Y),TS(T2)) = 7 //setting to max of write\_TS(Y)=3 or TS(T2)=7  
//write\_TS(Y) is now 7

T2: read\_item(X)  
write\_TS(X) < TS(T2) //check is good proceed  
execute read\_item(X)  
set read\_TS(X) ← max(read\_TS(X),TS(T2)) = 7 //setting to max of read\_TS(X)=3 or TS(T2)=7  
//read\_TS(X) is now 7

T2: write\_item(X)  
read\_TS(X) = TS(T2) && write\_TS(X) < TS(T2) //check is good proceed  
execute write\_item(X)  
set write\_TS(X) ← max(write\_TS(X),TS(T2)) = 7 //setting to max of write\_TS(X)=3 or TS(T2)=7  
//write\_TS(X) is now 7

Result: Schedule F completes successfully.