Exercises on Time-Stamping

Consider a DBMS without concurrency control and the following transaction history (where BoT means Beginning of Transaction, R= Read, RU= Read for Update, W= Write; the acc# will be used to refer to each action):

Acc#	**T1**	**T2**	**T3**
10		BoT	
20		R(D)	
30		RU(E)	
40			BoT
50			RU(A)
60			RU(C)
70			W(C)
80			W(A)
90		W(E)	
100		RU(B)	
110	BoT		
120	RU(F)		
130	W(F)		
140	R(A)		
150		W(B)	
160	R(E)		
170			R(F)
180			W(A)
190			RU(B)
200			W(B)
210		Commit	
220			Commit
230	Commit		

Suppose now that we implement a basic *time-stamping* concurrency control. How does this history would look like? For each abort performed (if any) think whether, according to this history, it was really necessary or not.
Would it change if we assume an extended time-stamping version with recoverability means? Do you think that dynamic time-stamping could help to make it more efficient? (i.e., avoid unnecesary aborts).

Exercises on Time-Stamping

Consider a DBMS without concurrency control and the following transaction history (where BoT means Beginning of Transaction, R= Read, RU= Read for Update, W= Write; the acc# will be used to refer to each action):

#Acc	T1	T2	T3
10			BoT
20		BoT	
30	BoT		
0		R(E)	
50	R(A)		
60	W(A)		
70			R(A)
80			W(A)
90	R(F)		
100	R(D)		
110	R(E)		
120	W(E)		
130		R(C)	
140		W(C)	
150		R(E)	
160			R(F)
170			W(F)
180		COMMIT	
190	COMMIT		
200			COMMIT

Suppose now that we implement a basic *time-stamping* concurrency control. How does this history would look like? For each abort performed (if any) think whether, according to this history, it was really necessary or not.
Would it change if we assume an extended time-stamping version with recoverability means? Do you think that dynamic time-stamping could help to make it more efficient? (i.e., avoid unnecesary aborts).