Given Name: ..... Family name: .....

1) (40%) Given a DBMS without any concurrency control mechanism, let's suppose that we have the following history (actions have been numbered just to facilitate referencing them):

#Acc	T1	T2	T3
10			BoT
20		BoT	
30	BoT		
40		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(C)
170			W(C)
180		COMMIT	
190	COMMIT		
200			COMMIT

Let's suppose now that the DBMS is based on a **dynamic timestamping** technique. Briefly explain how would result the same history? **Is any transaction cancelled?**

.....
.....
.....
.....
.....
.....

2) (30%) Supose you have to join two relations R(<u>A</u>,B) and S(<u>B</u>,C) that are sitted in two different machines, where B is a foreign key from R to S, and primary keys are underlined. All three attributes are integers, and R has double number of tuples than S. Is it worth to use a semi-join strategy to obtain R*S? Briefly justify why.

.....
.....
.....
.....
.....
.....

3) (30%) Explain what framework, whether MapReduce or Aggregation Framework, achieve a higher degree of parallelism when execting jobs. Justify your answer based on their global physical query optimiser and how they apply ordering to generate the physical access plans.