Homework 6 (70 Points)

As for the previous homework, submit one .zip file containing six text files named Q1.txt to Q3.txt, each containing the answer to the corresponding question. For each question about a schedule, provide the answer schedule in the first line and a one paragraph justification of your result afterwards. Use the following notation for schedules and specify all of the following operations once they arise:

- Rt(o) - transaction t reads object o
- Wt(o) - transaction t writes object o
- St(o) - transaction t requests a shared lock on object o
- Xt(o) - transaction t requests an exclusive lock on object o
- Ut(o) - transaction t releases locks on object o
- Ct - transaction t commits
- At - transaction t aborts

Q1) (20 Points) Specify a schedule (reads, writes, commits, aborts, locking and unlocking requests) without deadlock that could have been generated by non-conservative strict 2PL but not by conservative strict 2PL!

Q2) (20 Points) Give a schedule (reads, writes, commits, aborts, locking and unlocking requests) which avoids cascading aborts and could have been generated by non-strict 2PL but not by strict 2PL!

Q3) (30 Points) Someone proposes the following mechanism for proactively avoiding deadlocks: we associate each transaction T with the number N(T) of objects the transaction will maximally access (read or write). If transaction Ti requests a lock held by transaction Tj, Ti is aborted if N(Ti) < N(Tj) and can wait otherwise until the lock becomes available. Does this scheme work (i.e., avoid deadlocks in each case)? Justify.