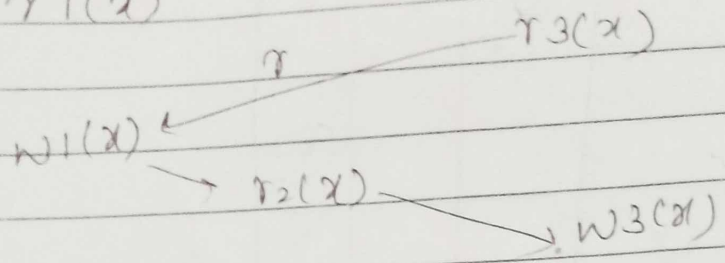
DSN Tut 14

Yash Agarwal 21104039

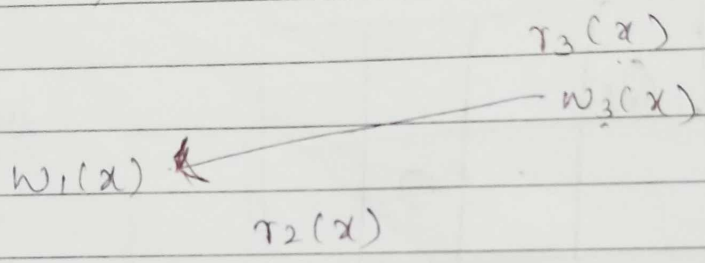
Q1

a)



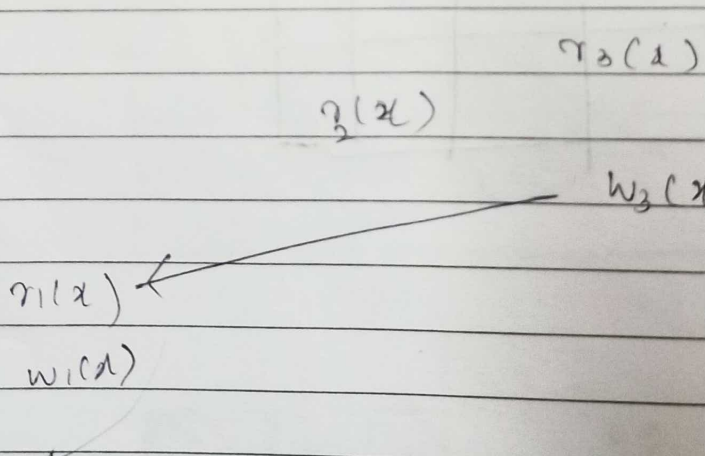
Not Serializable

b)



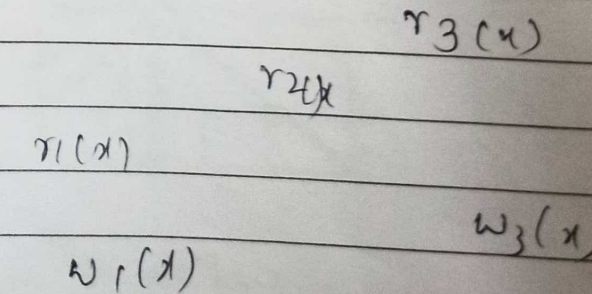
Not Serializable

c)



Serializable

d)

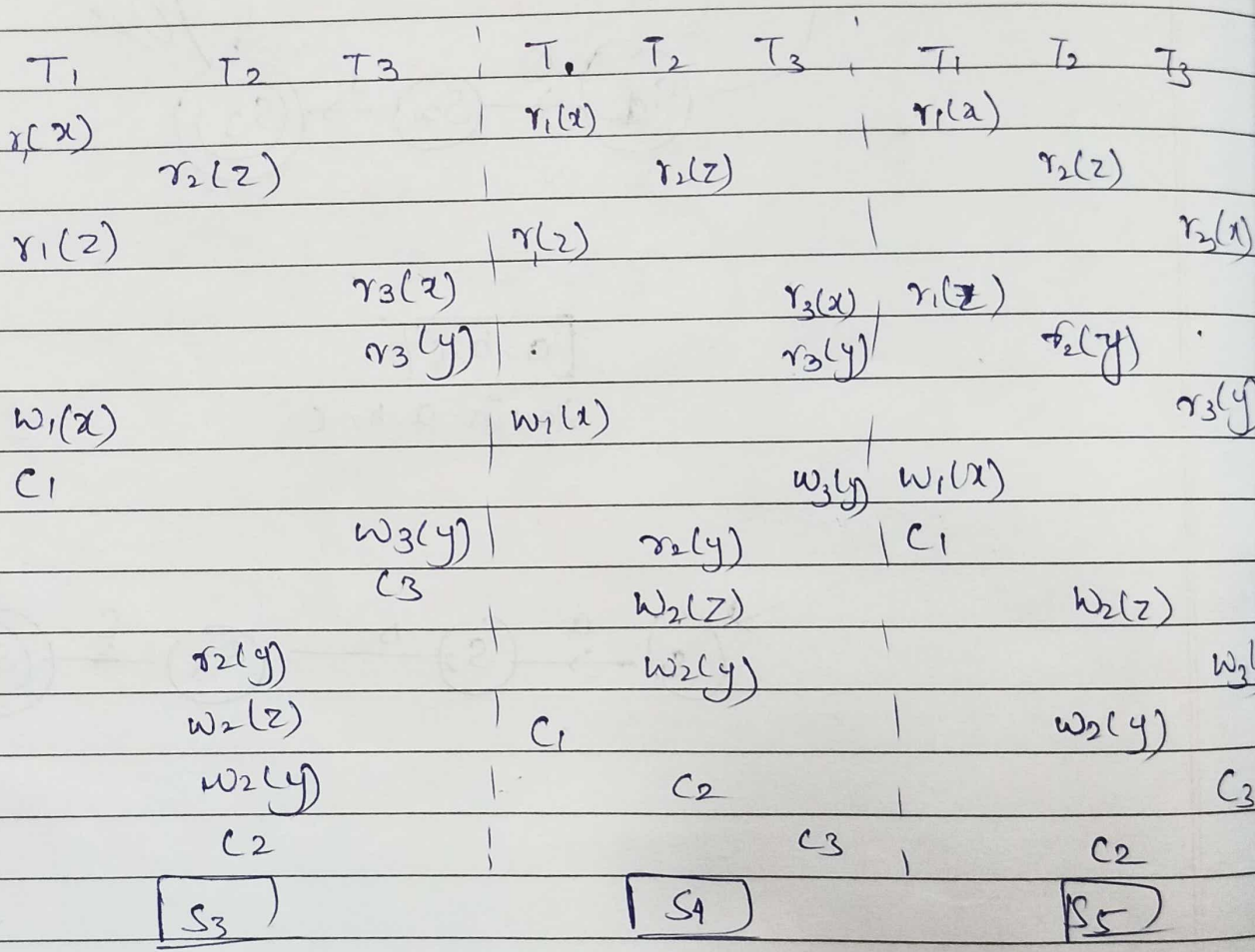


Not Serializable

Q2

S_3 : $r_1(x)$; $r_2(z)$; $r_1(z)$; $r_3(x)$ $r_3(y)$; $w_1(x)$; c_1 ; $w_3(y)$; c_3 ; $r_2(y)$; $w_2(z)$; $w_2(y)$; c_2 ;

S_4 : $\notin S_5$;



S_3 : recoverable, cascadeless, strict recoverable

S_4 : Not cascadeless, not recoverable

S_5 : recoverable, cascadeless, not strict recoverable

FREEMIND

Date

Page

Q3

(i) S_1 : T_1 has shared lock on x
T_2 has lower priority → it will be aborted
T_3 get exclusive - lock on y - T_3 now finished with, commits and releases all locks. T_1 arquired lock, procud & finishes. T_2 get started

S_2 : T_1 has shared lock on x T_2 has exclusive - lock on y. T_3 has exclusive lock on y. T_3 released the lock and commits. T1 has lock on y and procuds.

(ii) In deadlock detection, transaction are allowed to wait they are not aborted until a deadlock has been detected

S_1 : T_1 has shared lock on x. T_3 gets exclusive lock on y. T_3 finished, commit and relased locks. T_1 has exclusive lock on y. [No dead lock]

S_2 : There is a [deadlock]

(iii) S1 : with conservation and strict 2PL, the sequence is easy. T_1 acquired lock on both x & y commit, releases locks, then T_2 then T_3

S_2 : The sequence is easy. T_1 aquires lock on both x & y commits, releases locks, then T_2, then T_3