Q1.

Query 1: $\Pi_{sid, Name}(\sigma(\text{course.semester} \leq 6 \wedge \text{course.semster} \geq 4 \wedge \text{Enrolled.semester} \leq 3 \wedge 1 \leq \text{Enrolled.semester})(\text{Enrolled} \bowtie \text{student}) \bowtie \text{course}))$

Query 2.

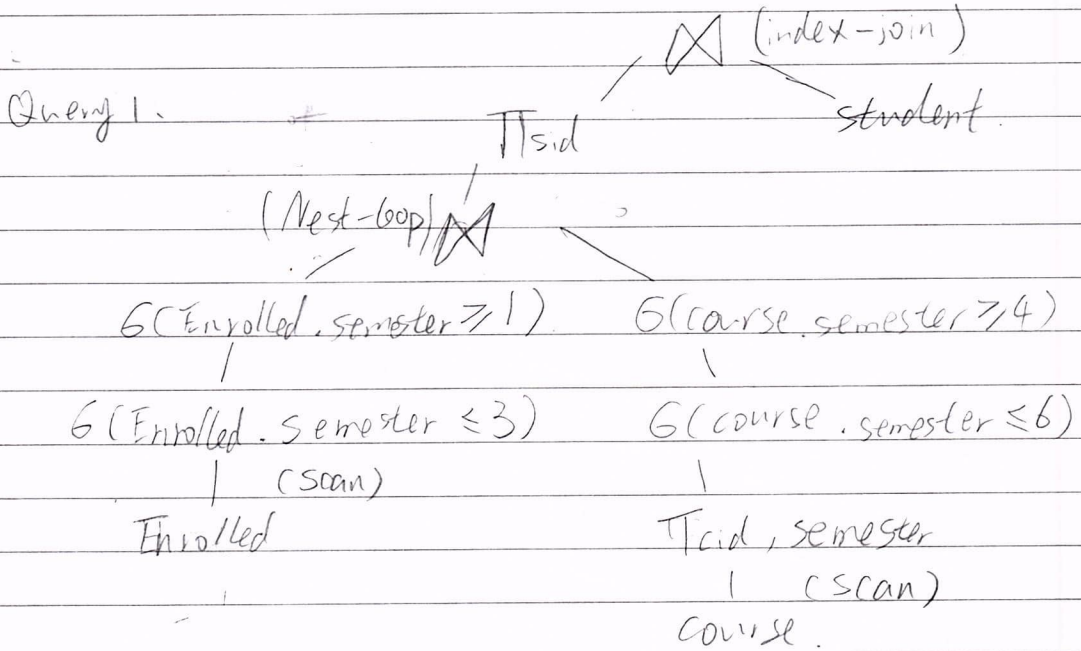
$PE_1 := \rho_{sid1, cid1, semester1, require1}(\text{Enrolled} \bowtie \text{prereq})$

$PE_2 := \rho_{sid2, cid2, semester2, require2}(\text{Enrolled} \bowtie \text{prereq})$

$\Pi_{sid1, Name1}(\Pi_{sid1, Name1, cid1}(PE_1) - (\Pi_{sid1, Name1, cid1}(\sigma_{sid1 = sid2 \wedge (require1 = cid2) \vee require1 = 'None'}(PE_1 \times PE_2))))$

Q2.

Query 1.



Schedules:

S: $r_1[x]$ $r_1[y]$ $w_1[x]$ $r_2[y]$ $w_3[y]$ $w_1[x]$ $r_2[y]$

we have conflict:

$r_2[y] \rightarrow w_3[y]$

$w_3[y] \rightarrow r_2[y]$

$r_1[y] \rightarrow w_3[y]$

It's impossible to fix first two confile in serial exction, hence S is not serializable.

recoverable:

S: $r_1[x]$, $r_1[y]$, $w_1[x]$, $r_2[y]$, $w_3[y]$, $w_1[x]$, c_1, $r_2[y]$, c_2.

conflict - serializable:

S: $r_1[x]$, $r_1[y]$, $w_1[x]$, $r_2[y]$, $w_3[y]$, c_3, $w_1[x]$, c_1, $r_2[y]$ c_2.

2. locking schedulers:

Assume we only have 2 transaction T_1 and T_2. and th will be not serializable only when conflict exist:

① T_1 write A, T_2 read/write A

② T_1 read/write A, T_2 write A.

let say the conflict set $S = \{C_1, C_2 \cdots C_n\}$, and assume $S_i \{C_1 \cdots C_i\}$ are in case 1 $S_i \{C_{i+1} \cdots C_n\}$ in case 2

then $\forall C \in S_1$, T_1 will acquire exclusive_lock(C_i)
T_2 will acquire exclusive/share_lock(C_i)

When T_1 sucessfully acquire an ex-lock(C_i) in S_1, T_2 have to acquire the lock of C_i in somewhere, then T_2 will waiting. Since we follow 2PL protocol, T_1 will not release ex-lock(C_i) until it required all other lock in S. If T_1 reach a lock that T_2 already lock before T_2 waiting, then dead-lock exist. Otherwise, T_1 acquire all the lock in S successfully, which mean T_2 doesn't mutate any of them yet (because T_2 also follow 2PL protocol) Then T_2 actully access all the C_i in S after T_1, it's same as a serial execution.

In case ②, it's the same when T_2 acquire a ex_lock(C_i) in S_2.