Consider the relation \texttt{Class}(\texttt{dept, cnum, title}). The tuple \('CS', 143, 'Database'\) in the relation means that department \('CS'\) is offering class 143 on 'Database'.

Write a relational algebra expression that returns all departments that offer at least three classes. Remember that simplicity and clarity count as well as correctness.

\[
\pi_{A.dept}(\sigma_{A.dept=B.dept\land A.cnum<>B.cnum\land A.cnum<>C.cnum\land B.cnum<>C.cnum}(\rho_{A}(\text{Class}) \times \rho_{B}(\text{Class}) \times \rho_{C}(\text{Class})))
\]

Consider the relation \texttt{Enroll}(\texttt{sid, dept, cnum}). The tuple (301, 'CS', 143) in the relation means that student 301 is taking class 143 offered by the CS department. Assume that \texttt{sid} is the key of students and (\texttt{dept, cnum}) is the key of classes.

Write a relational algebra expression that returns all pairs of students who are taking at least one Electrical Engineering class (\texttt{dept='EE'}) in common, but not any common Computer Science classes (\texttt{dept='CS'}). Make sure that you include each pair only once — for example, if your expression returns (301, 405), then it should not also return (405, 301). Remember that simplicity and clarity count as well as correctness.

\[
\pi_{E1.sid,E2.sid}(\sigma_{E1.sid=E2.sid\land E1.dept='EE'\land E1.cnum=E2.cnum}(\rho_{E1}(\text{Enroll}) \times \rho_{E2}(\text{Enroll}))) - \\
\pi_{E1.sid,E2.sid}(\sigma_{E1.sid=E2.sid\land E1.dept='CS'\land E1.cnum=E2.cnum}(\rho_{E1}(\text{Enroll}) \times \rho_{E2}(\text{Enroll})))
\]

The expression \(\sigma_C(R) - S\) is always equivalent to \(\sigma_C(R) - \sigma_C(S)\), where \(C\) can be any condition.

\[\text{TRUE}\].

The expression \(\sigma_{C \cup D}(R \bowtie S)\) is equivalent to \([\sigma_C(R) \bowtie S] \cup [R \bowtie D(S)]\) for bags and sets (Condition \(C\) is only over \(R\) attributes; condition \(D\) is only over \(S\) attributes.)

\[\text{FALSE}.\ \text{Bag semantics may introduce duplicates when taking the UNION}.
\]