1. (30 marks) For each of the following, state whether the statement is true or false. If it is false, correct the statement without changing the underlined text. If the statement is true, do not write anything more, just indicate true.

(a) Persistence means that the data can never be deleted.
   F means the data will remain on disc after the program that created it has terminated.

(b) Object identity is said to be immutable, which means it doesn’t speak.
   F it is not susceptible to change, i.e. cannot be changed by the programmer.

(c) Fragmentation transparency in a centralized relational database means that the user does not need to be aware that the tables are replicated.
   F in a distributed database ..... fragmented.

(d) XML database systems allow collections of XML data to be stored and queried.
   T

(e) For horizontal fragmentation, the disjointness condition means that the fragments should have disjoint columns.
   F rows

(f) Vertical fragmentation divides a relation into fragments using the relational algebra project \((\pi)\) operator.
   T

(g) Relations fragmented using derived horizontal fragmentation can be put back together using the relational algebra join \((\bowtie\bowtie)\) operator.
   F union \((\cup)\)

(h) The extended ER model is the original ER model with the addition of inheritance.
   T

(i) The term “ISA” arises when talking about aggregation.
   F inheritance

(j) Distributed relational databases are good for companies which have a need for data with complex structure and complex operations.
   F have a need for data and processing in many geographically dispersed locations.

(k) Closure is a property of a data model.
   F query language.

(l) Impedance mismatch is an issue which arises when designing the interactive API of a query language.
   F arises with queries embedded in a programming language.

(m) In object-oriented database querying, to know two set objects are shallow equal means that all we need to know is that they have the same values in their attributes.
   F that the sets have the same cardinalities and the elements are pairwise identical.
(n) **XPath queries** can create results which have a different “shape” from the input document(s).  
\[ \text{F cannot create new shapes.} \]

(o) **OQL**, the query language for OODBs, has the “//” operator which allows one to search for a subobject at an arbitrary level of nesting.  
\[ \text{F does not have any such operator.} \]

(p) The “just keep what you need” heuristic, which is part of optimizing the order of operations for relational algebra queries, is achieved by introducing new select \((\sigma)\) operators.  
\[ \text{F projection \((\pi)\)} \]

(q) **Natural join** is both commutative and associative.  
\[ \text{T} \]

(r) For distributed relational databases, algebraic query optimization takes the fragmentation structure into account.  
\[ \text{T} \]
2. (8 marks) Consider the following relations for a relational database:

- Students(StudID, Name, Year) Primary key is {StudID}
- Courses(CNo, CName, Desc) Primary key is {CNo}
- Taken(StudID, CNo, Term, Mark) Primary key is {StudID, CNo}

(a) Give the relational algebra expression for the fragment of the Students relation for students in year 4.

\[ F_1 = \sigma_{\text{year}=4} \text{(Students)} \]

(b) Looking at your answer to part (a), what other fragment(s) needs to be present to guarantee reconstruction of the Students relation.

\[ F_2 = \sigma_{\text{year}\neq 4} \text{(Students)} \]

(c) Give the relational algebra expression for that fragment of the Taken relation where the student taking the course is in year 4. Express this in terms of your answer to part (a).

\[ F_3 = \text{Taken} \bowtie F_1 \]

(d) Looking at your answer to parts (b) and (c), what other fragment(s) must be present to guarantee reconstruction of the Taken relation?

\[ F_4 = \text{Taken} \bowtie F_2 \]

(e) Give the relational algebra expression to perform the reconstruction of relation Taken from the fragments in parts (b) and (d).

\[ F_3 \cup F_4 \]

(f) (3 marks) For privacy reasons, information concerning the Marks should not be released to certain university staff. Give relational algebra expressions for two fragments of Taken, one which can be publicly released, and one which should not. Also give the expression which reconstructs the original Taken relation from these fragments.

Private = \[ \pi_{\text{StudID,CNo,Mark}} \text{(Taken)} \]

Public = \[ \pi_{\text{StudID,CNo,Term}} \text{(Taken)} \]

reconstruct = Private \bowtie◁ Public
3. (8 marks) Which of the following pairs of relational algebra expressions give “the same” answer, i.e. are what we would call equivalent? If one or both of the expressions are undefined, then mark “not equivalent”.

Assume the relations being used are $S(s, n, y)$, $C(c, a, d)$ and $T(s, c, t, m)$ (these are the relations from the previous question, reduced to single letters).

(a) $C \bowtie T$ not equivalent
   and $C \times T$ (different numbers of attributes)

(b) $(S \bowtie C) \bowtie T$ equivalent
   and $(S \bowtie T) \bowtie C$

(c) $\pi_c (C \cap T)$ (not defined) not equivalent
   and $\pi_c(C) \cap \pi_c(T)$

(d) $\sigma_{n=“Jane”}$ and $m = 85$ $(S \bowtie T)$ not equivalent
   and $\sigma_m = 85$ $(S) \bowtie \sigma_{n=“Jane”}(T)$ (wrong attributes for the relations)

(e) $\pi_{s,n}(\sigma_{n=“Jane”}$ and $m = 85$ $(S \bowtie T)$ equivalent
   and $\pi_{s,n}(\sigma_m = 85 (T)) \bowtie \pi_{s,n}(\sigma_{n=“Jane”}(S))$

(f) $\pi_{s,n,c}(\sigma_{n=“Jane”}$ and $m = 85$ $(S \bowtie T) \bowtie C)$ not equivalent
   and $\pi_{s,n,c}(\sigma_{n=“Jane”}$ and $m = 85$ $(S \bowtie C) \bowtie (\pi_{s,m}(T)))$ (need to keep c from T)

(g) $S \bowtie T$ equivalent
   and $\pi_{s,n,y}(S \bowtie T)$

(h) $\sigma_y = 4$ $(S \bowtie T)$ not equivalent
   and $S \bowtie \sigma_y = 4 (T)$ (y is not in T)
4. (12 marks) Consider the following relations for data for a relational database:

- Car(VIN, make, year) Primary key is \{VIN\}
- Driver(DrID, Name, CreditRating) Primary key is \{DrID\}
- TestDrives(VIN, DrID, date) Primary key is \{Vin, DrID\}

Suppose the following fragments have been created (we intend to store the first and third fragments at one site, and the second and fourth fragments at another):

- \(\text{SUVs} = \sigma_{\text{make} = \"SUV\"}(\text{Car})\)
- \(\text{Non-SUVs} = \sigma_{\text{make} \neq \"SUV\"}(\text{Car})\)
- \(\text{DroveSUV} = \text{Driver} \bowtie \text{SUVs}\)
- \(\text{DroveNonSUV} = \text{Driver} \bowtie \text{Non-SUVs}\)

*there is a problem here as the semijoins don’t make sense. TestSUVs = TestDrives \bowtie \text{SUV’s} and TestNonSUVs = TestDrives \bowtie \text{Non-SUV’s} should have been defined, but not even this would have helped the query.

(a) (2 marks) Suppose we have the SQL query:

```
Select Name, VIN, date
From Driver as D, TestDrives as T
where D.CreditRating > 2 and D.DrID = T.DrId
```

Translate this query to relational algebra.

\[
\pi_{\text{Name, VIN, date}}(\sigma_{\text{Driver.CreditRating} > 2 \text{ and Driver.DrID = TestDrives.DrID}}
(Driver \times \text{TestDrives}))
\]

Note: we don’t know yet that this is a join, so Cartesian Produce is used.

(b) (3 marks) Show the initial query tree corresponding to your algebra query just above.

![Query Tree Image]
(c) (7 marks) Replace any relations in your tree with the fragments defined above, expressed as qualified relations of the form [R:F] where F contains any predicates you know to be true about the fragment. After doing this, perform any further optimizations possible on your algebra tree.

Driver gets replaced by [DroveSUF: make = “SUV”] ∪ [DroveNonSUV: make ≠ “SUV”] (which doesn’t really help with optimization). As given, TestDrives has no fragments. I was looking for a ∪ at the top of the tree, and the fact that the σ_{CreditRating>2} got pushed down to near the Driver fragments.
5. (12 marks) Answer any **three** of the following (only the first three answers will be marked):

(a) What are some of the reasons for looking at orthogonal design choices, such as we saw with the design of the directory structure for a distributed database?

(b) Explain the reconstruction condition for fragmentation in a distributed database and why it is needed.

(c) We said that aggregation and inheritance are two orthogonal design issues for object-oriented databases. Is the consideration of collections also orthogonal to these other two issues? Explain why or why not.

(d) Explain what bag operations like bag union and bag intersection are and why SQL systems need to worry about them.

(e) Explain what a database schema is and justify (or refute) the statement that XML documents carry their schema around with them.

(f) Explain the basic idea behind algebraic query optimization of SQL queries and why it is important.

(g) Show by a counterexample that projection does not distribute over set difference.
For rough work or answers to question 5.