Exercise 1 Containment (20 points)
Decide for the pairs of queries below whether \( q \subseteq q' \). Prove that containment holds using the canonical database method, or provide a database instance that contradicts containment.

1. \( q(x) : \neg R(x, y), R(y, z), R(z, x) \) \( q'(x) : \neg R(x, y), R(y, z), R(z, u), R(u, v), R(v, z) \).
2. \( q(x, y) : \neg R(x, u, u), R(u, v, w), R(w, w, y) \) \( q'(x, y) : \neg R(x, u, v), R(v, v, v), R(v, w, y) \).

Exercise 2 Datalog (20 points)
A positive Boolean tree has leaves labeled with 0 or 1, and internal nodes which are operators AND, OR. The root of a boolean tree returns 0 or 1 by computing from the leaves upwards in the obvious way. We can represent a binary positive boolean tree using the following stored (EDB) predicates:

- \( \text{Leaf0}(x) \): a unary table containing the identifiers of each leaf labeled 0. (Alternatively, you can think of \( \text{Leaf0}(x) \) as a logical predicate that is true of node \( x \) if \( x \) is a leaf labeled 0.)
- \( \text{Leaf1}(x) \): a unary table containing the identifiers of each leaf labeled 1.
- \( \text{And}(x, y_1, y_2) \): a ternary table containing tuples \((x, y_1, y_2)\) if \( x \) is an AND node whose children are nodes \( y_1 \) and \( y_2 \).
- \( \text{Or}(x, y_1, y_2) \): a ternary table containing tuples \((x, y_1, y_2)\) if \( x \) is an OR node whose children are nodes \( y_1 \) and \( y_2 \).
- \( \text{Root}(x) \): a unary table containing the root node.

1. Write a datalog query that computes the boolean relation \( \text{Answer()} \) which is true if and only if the root node returns 1.
2. Is this query expressible in UCQ, i.e. as a union of conjunctive queries?

Exercise 3 Updating views (10 points)
Given two tables \( \text{Students}(\text{sid}, \text{name}, \text{age}) \) and \( \text{Enrolled}(\text{studid}, \text{cid}, \text{grade}) \) consider the view \( V \) defined as follows:

\[
V(n, s, c) : \neg \text{Students}(s, n, a) \& \text{Enrolled}(s, c, g) \& g = "B"
\]

The view result consists of tuples of the form \((\text{name}, \text{sid}, \text{cid})\) and suppose \( t=\text{(Joe, 8250, 445)} \) is an example tuple in the result. For each of the following update operations on \( V \), describe how they can be translated into operations on \( \text{Students} \) and/or \( \text{Enrolled} \), and any complications involved.

1. Insert a new tuple \((\text{Mary, 8251, 645})\) into \( V \).
2. Delete tuple \( t \) from \( V \).
Exercise 4 Output of join algorithms (12 points)
Suppose we have two unary (one attribute only) relations, R and S:

\[
\begin{array}{cc}
R & S \\
7 & 8 \\
2 & 4 \\
9 & 2 \\
8 & 1 \\
3 & 3 \\
9 & 2 \\
1 & 7 \\
3 & 3 \\
6 & 3 \\
\end{array}
\]

Show the result of joining R and S using each of the following algorithms. List the results in the order that they would be output by the join algorithm. Note that the result relation contains only one attribute, which is the common attribute between R and S.

1. Nested loops join (one tuple at a time). Use R for the outer loop and S for the inner loop.

2. Sort merge join.

3. Hash join. Assume there are two hash buckets, numbered 0 and 1, and that the hash function sends even values to bucket 0 and odd values to bucket 1. In Phase II, use R as the “build” relation and S as the “probe” relation. Assume that bucket 0 is read first and that the contents of a bucket are read in the same order as they were written.

Exercise 5 Costs of join algorithms (28 points)
Consider the join \( R \bowtie_{a=A} S \), given the following information about the relations to be joined. The cost metric is the number of page I/Os unless otherwise noted, and the cost of writing out the result should be uniformly ignored.

Relation R contains 10,000 tuples and has 10 tuples per page.
Relation S contains 2,000 tuples and also has 10 tuples per page.
Attribute b of relation S is the primary key for S.
Both relations are stored as simple heap files.
Neither relation has any indexes built on it.
52 buffer pages are available.

1. What is the cost of joining R and S using a page-oriented simple nested loops join? What is the minimum number of buffer pages required for this cost to remain unchanged?

2. What is the cost of joining R and S using a block nested loops join? What is the minimum number of buffer pages required for this cost to remain unchanged?

3. What is the cost of joining R and S using a sort-merge join? What is the minimum number of buffer pages required for this cost to remain unchanged?

4. What is the cost of joining R and S using a hash join? What is the minimum number of buffer pages required for this cost to remain unchanged?
5. What would be the lowest possible I/O cost for joining R and S using any join algorithm, and how much buffer space would be needed to achieve this cost? Explain briefly.

6. How many tuples does the join of R and S produce, at most, and how many pages are required to store the result of the join back on disk?

7. Would your answers to any of the previous questions in this exercise change if you were told that R.a is a foreign key that refers to S.b?

**Exercise 6 Improvement of hash join (20 points)**

Consider a join of relations R and S using hash join, where \( M = \# \text{pages in } R = 400 \text{ pages} \) and \( N = \# \text{pages in } S = 500 \text{ pages} \). Assume that we have \( B = 40 \text{ pages} \) of memory available in our buffer. We notice that there is more than enough memory, so we keep one of the hash buckets in memory to avoid writing it during the partitioning phase and then re-reading it during the probing phase. Assuming that all buckets have the same size.

1. How many buckets should be used in order to still be able to perform the join in two passes?

2. What would be the cost in number of I/Os of the operation proposed in part 1?