Solution Report

1 Implementation Details

I implemented the hash join algorithm in Node.js to evaluate the query $q(A, B, C) : -R_1(A, B), R_2(B, C)$. The code can be found in `Solution.js`.

The implementation steps are as follows:

1. **Data Structures**: I represented relations R_1 and R_2 as arrays of objects, where each object represents a tuple with named attributes (e.g., { `A: 1, B: 10` }).
2. **Hash Phase**: I created a hash map (using a JavaScript `Map`) for relation R_2.
   - The key for the hash map is the join attribute `B`.
   - The value is a list of tuples from R_2 that have that specific value for `B`. This handles cases where multiple tuples in R_2 share the same join key.
3. **Probe Phase**: I iterated through each tuple in relation R_1.
   - For each tuple $t_1 \in R_1$, I extracted the value of attribute `B`.
   - I probed the hash map with this key.
   - If a match was found, I iterated through the list of matching R_2 tuples and produced a result tuple (A, B, C) for each match.

2 Correctness

The algorithm is correct because it faithfully implements the definition of a natural join:

- It finds all pairs of tuples (t_1, t_2) such that $t_1 \in R_1$, $t_2 \in R_2$, and $t_1.B = t_2.B$.
- By iterating through all of R_1 and looking up all matching tuples in R_2 via the hash map, we ensure that no valid join combinations are missed.
- The hash map allows for efficient $O(1)$ average time complexity lookups for each tuple in R_1, making the overall complexity roughly $O(|R_1| + |R_2|)$ (assuming the output size is not dominated by a cartesian product).

3 Execution Results

I ran the algorithm with a dataset of 10 tuples for R_1 and 10 tuples for R_2.

Relation $R_1(A, B)$

A	B
1	10
2	20
3	30
4	40
5	50
6	60
7	70
8	80
9	90
10	100

Relation $R_2(B, C)$

B	C
10	100
20	200
30	300
40	400
50	500
10	101
60	600
70	700
80	800
110	1100

Join Results $q(A, B, C) : -R_1(A, B), R_2(B, C)$

A	B	C
1	10	100
1	10	101
2	20	200
3	30	300
4	40	400
5	50	500
6	60	600
7	70	700
8	80	800

Note:

- Tuples with $B = 90$ and $B = 100$ in R_1 did not find a match in R_2.
- The tuple with $B = 110$ in R_2 did not find a match in R_1.
- The value $B = 10$ appeared twice in R_2, resulting in two join tuples for $A = 1$.