Part B **(50 points)**

1. **(8 points)** A generic SQL query with aggregation but with no `WHERE`, `GROUPBY`, or `HAVING` clause is:

   ```
   SELECT aggr(A) FROM R
   ```

   where aggr is an aggregate function, `R` is a relation, and `A` is an attribute of `R`. Consider a basic relational algebra with operators σ, Π, $\times$, and $-$, and attribute or relation renaming. Are there any standard SQL aggregate functions for which a query equivalent to the SQL query above can be formulated in this relational algebra? If so, show the equivalent algebra expression(s).

2. **(16 points)**

   (a) Draw a rough architectural diagram of a simple centralized database management system, illustrating at least a file & buffer manager, data manager, query optimizer, query execution engine, transaction manager, disk(s), buffer, lock table, and user/application interface. Include arrows in your diagram labeled with the information that passes between components. You need not be enormously detailed—the purpose is to exhibit an understanding of basic DBMS architecture, and to set up for parts (b) and (c).

   (b) Modify, augment, or mark your diagram from part (a) to delineate the client-server architecture for a typical relational DBMS. Note that relational DBMS's usually use "query shipping" rather than "data shipping" in client-server environments.

   (c) Modify, augment, or mark your diagram from part (a) to delineate the client-server architecture for a typical object-oriented DBMS. Note that object-oriented DBMS's usually use "data shipping" rather than "query shipping" in client-server environments.

3. **(4 points)** In a relational DBMS, when *query precompilation* (sometimes called *stored-compiled SQL*) is used, a query is compiled once into an execution plan, then the plan is stored in the database and used whenever the query is issued. It is a big savings to avoid compilation each time a query is issued, since query compilation is typically very time consuming. Unfortunately precompiled queries may execute with poor performance. Briefly (in 1–3 sentences) explain why.

4. **(6 points)** Consider the execution of a multiway join $R_1 \bowtie R_2 \bowtie \cdots \bowtie R_n$ such that:

   - Either a *left-deep* or a *right-deep* (but not a *bushy*) query execution plan is used.
   - Each constituent 2-way join in the query execution plan uses a simple hash-based ("build and probe") join method.
   - By convention, the left operand is the "build" relation and the right operand is the "probe" relation.
   - Composition of operations during query execution is achieved using an *iterator* approach.

   How many temporary hash tables need to exist at one time for the left-deep query execution plan? For the right-deep plan?

5. **(16 points)** In an "advanced" DBMS, at least four distinct types of *rules* may be used in the system or provided to the user. Give one simple example of each type of rule below, along with a brief (1–3 sentence) explanation of what it does.

   (a) Deductive database rule

   (b) Active database rule (or trigger)

   (c) Query rewrite (or transformation) rule

   (d) Query plan generation (or implementation) rule