The following problems use these table definitions:

```sql
CREATE TABLE Person (
    pid INT PRIMARY KEY, -- person ID
    name VARCHAR(100));   -- person name

CREATE TABLE Email (
    eid INT PRIMARY KEY,       -- email ID
    pidFrom INT REFERENCES Person, -- email sender
    length INT);                  -- email char length

CREATE TABLE EmailTo (
    eid INT REFERENCES Email,   -- email ID
    pidTo INT REFERENCES Person,  -- email recipient
    PRIMARY KEY (eid, pidTo));
```

Relational algebra operators:

- Union \( \cup \)
- Difference \(-\)
- Selection \(\sigma\)
- Projection \(\pi\)
- Join \(\bowtie\)
- Rename \(\rho\)
- Duplicate elimination \(\delta\)
- Grouping and aggregation \(\gamma\)
- Sorting \(\tau\)

A **warmup**: Find the length of each email that Alice sent.

```sql
SELECT E.length
FROM Person P, Email E
WHERE P.pid = E.pidFrom AND
    P.name = 'Alice';
```

Draw the RA tree for the query
The table declarations are repeated for your convenience:

CREATE TABLE Person
   (pid  INT PRIMARY KEY, -- person ID
    name VARCHAR(100)); -- person name

CREATE TABLE Email
   (eid     INT PRIMARY KEY,       -- email ID
    pidFrom INT REFERENCES Person, -- email sender
    length  INT);                  -- email char length

CREATE TABLE EmailTo
   (eid   INT REFERENCES Email,   -- email ID
    pidTo INT REFERENCES Person,  -- email recipient
    PRIMARY KEY (eid, pidTo));

Relational algebra operators:
Union ∪ Difference — Selection σ Projection π Join ⋈ Rename ρ Duplicate elimination δ Grouping and aggregation γ Sorting τ

An extended problem: Find the number of emails that each person has received.

SELECT P.name, COUNT(*)
   FROM Person P, EmailTo T
   WHERE P.pid = T.pidTo
   GROUP BY P.pid, P.name;

Draw the RA tree for the query