1. List the members of your group below. Underline your name.

2. Provide expressions for the minimum and maximum cardinalities of the result of each of the basic operators of the extended bag algebra as a function of the cardinalities of its operands. Justify your answers.

3. Provide standard SQL (or closest possible) expressions of the *best-profit* and *packet-grouping* queries described in Section 1.1 of the *AQuery* paper.[1]

[1]Alberto Lerner and Dennis Shasha, "AQuery: Query Language for Ordered Data, Optimization Techniques, and Experiments," in *Proceedings of the 29th International Conference on Very Large Data Bases (VLDB)* (Berlin, Germany, 2003).

4. Recall the SQL query about TAs from the previous exercises *the names and IDs of the TAs who are the TAs of the maximum number of students for r credits, for each distinct value of r occurring in the database). Comment on the correctness of the following solution by Anthony Naddeo, and suggest improvements or alternative expressions of the query.

```
create table students(
  id serial primary key,
  name text,
  year integer );
create table courses(
  id serial primary key,
  title text,
  ta integer references students(id) );
create table enrolls(
  student integer references students(id),
  courses integer references courses(id),
  credits integer );
-- Shows all students that all tas are responsible for
CREATE VIEW ta_students AS
  SELECT c.ta, s.id, e.credits
  FROM courses c, students s, enrolls e
  WHERE s.id = e.student AND e.courses = c.id;
-- Shows just the total credits that each ta is resonpsible for
CREATE VIEW credits_responsible_for AS
  SELECT ta, sum(credits) AS num_credits
  FROM ta_students
  GROUP BY ta;
-- Shows just the total students that each ta is responsible for
CREATE VIEW students_responsible_for AS
  SELECT ta, count(id) AS num_students
  FROM ta_students
  GROUP BY ta;
-- Shows both of the top two tables as one
CREATE VIEW all_responsible AS
  SELECT ta, num_students, num_credits
  FROM students_responsible_for NATURAL JOIN credits_responsible_for;
-- Shows all distinct total-credit values with the max(cnt(students))
-- for each value
CREATE VIEW distinct_credits AS
  SELECT max(num_students) AS num_students, num_credits
  FROM all_responsible
  GROUP BY num_credits;
-- Shows the ta and max(cnt(sudents)) for each distinct credit sum
CREATE VIEW answer AS
  SELECT ta, num_students, num_credits
  FROM distinct_credits natural join all_responsible;
```

5. Provide an algebra equivalent of the query of Question 4.