Description
You are given a subset of SQL consisting of:

- Relation definitions of the form:

  CREATE TABLE Table
  (A_1 INTEGER,
   ...
   A_k INTEGER)

  Additionally, one attribute is designated as (column-based) PRIMARY KEY. For this attribute, a dense index should be built.

- DROP TABLE Table.

- single-column index definitions of the form:

  CREATE INDEX I ON Table(A).

- DROP INDEX I.

- SQL2 single-tuple INSERT commands.

- Queries of the following form:

  SELECT List of Attributes
  FROM Table
  ORDER BY F(A_1, ..., A_n) DESC
  LIMIT K

  where

  - List of Attributes contains some attributes of Table, and
  - F(A_1, ..., A_n) is a monotone function defined by a linear arithmetic expression with positive integer coefficients over the attributes, for example 11 * B + 2 * C in a schema that has A, B, and C as attributes. The general form of such an expression is:

    \[ \sum_{i=1}^{n} c_i * A_i \]

    where \( c_i > 0 \) for \( i = 1, \ldots, n \).

For each such query, a dense index over each attribute \( A_1, \ldots, A_n \) should be built.
You are supposed to implement:

1. A parser for the input (you can use a parser generator like javacc).

2. A query evaluator for the above subset of SQL. The evaluator should implement the version of the Threshold Algorithm which uses primary/secondary indexes. Also, the evaluator should use the file/index interface built in Project #1. Appropriate conversions between integers and strings should be made.

**Extra credit**

You may get up to 6% of the final grade for implementing one or more of the following:

- A general `WHERE` clause;

- An experimental comparison of the above approach with the naive approaches. Use a real-world dataset.