The main task in this assignment is implementing the normalization algorithms for 3NF, BCNF, and 4NF as described in the textbook and in class. The desired program, called `dbnorm`, reads schema and commands from standard input and writes responses to standard output. Please refer to the earlier assignments for general instructions, including those for packaging and submission, and use the class newsgroup for clarifications and discussions.

Input and Output The input consists of a sequence of semicolon-terminated *statements*. Each input statement elicits an output from the program. The input statements and their outputs are as follows:

- A *relation signature* declaration, composed of the relation name followed by its parenthesized, comma-separated attribute names.
  Examples: `R(A, B, C); Books(isbn, author, title, publisher, editor);`
  The output is the *canonical form* (described later) of the relation signature declared by the statement.
  Examples: `R(A, B, C);\nBooks(isbn, author, title, publisher, editor);\n`, where `\n` denotes a newline character.
- A *functional dependency* declaration, composed of one or more comma-separated attribute names followed by a right-arrow token `->` followed by one or more comma-separated attribute names.
  Examples: `A,B -> C; isbn -> title, publisher;`
  The output is the canonical form of each functional dependency. Examples: `A, B -> C;\nisbn -> title, publisher;\n`.
- A *multivalued dependency* declaration, similar to the functional dependency declaration but with a double-arrow token `->>` separating the left and right sides.
  Examples: `B ->> C; isbn ->> title, author;`
  The output is the canonical form of each multivalued dependency. Examples: `B ->> C;\nisbn ->> title, author;\n`.
- A *show* statement, for displaying the current schema, composed of the the literal `show;`.
  The output is the current schema in canonical form. For example, a show statement immediately following the two relation signature and four dependency statements above yields `Books(isbn, author, title, publisher, editor);\nR(A, B, C);\nA, B -> C;\nB ->> C;\nisbn -> title, publisher;\nisbn ->> title, author;\n`.
- *Normalization commands* for converting the current schema into 3NF, BCNF, and 4NF, and one for resetting the schema to the original one. These four commands are input as the literals `to3NF;`, `toBCNF;`, `to4NF;`, and `toOrig;`, respectively. During normalization, decomposing a relation `R` yields relations named `R1` and `R2`. If `R1` is decomposed further, it yields `R11` and `R12`, and so on.
  Each of these commands produces only a single blank line as output.
- A *trace* statement of the form `trace on;` or `trace off;`, for turning program tracing on or off. By default, program tracing is off. When program tracing is on, the

normalization commands produce extra output to show all intermediate steps in the transformation. Use the class newsgroup for further details.

Each trace command produces only a single blank line as output (but potentially modifies the outputs of commands to follow).

All relation and attribute names follow the lexical conventions of Java identifiers. Whitespace is optional everywhere in the input.

Disambiguation If an attribute name appears in multiple relations of the current schema, then any use of that name in a dependency is disambiguated by prefixing it with the appropriate relation name followed by a period. Attribute names in the input may be prefixed with relation names even when such prefixes are not required for disambiguation. For example, if the earlier schema is augmented with a relation `S(B,C,D,E)` then the functional dependency for `R` must be written as `A, R.B -> R.C;` or `R.A, R.B -> R.C;` but not as `A, B -> R.C;` (although in principle one could disambiguate `B` based on the context provided by `A`).

Canonical forms The canonical form of signatures, dependencies, and schemas follows closely the input format. The canonical form of a relation signature has a single space character following each comma. For example, the canonical form of `␣T␣(P,Q,␣R);` is `T(P,␣Q,␣R);`, where we use ␣ to make spaces explicit. Similarly, the canonical form of a dependency, functional or multivalued, has a single space following each comma and a single space on either side of the arrow token. The canonical form of a schema is a list of the canonical forms of the relations in the schema, one per line, followed by a list of the canonical forms of the dependencies, also one per line. Both lists are sorted lexicographically using the 7-bit ASCII collating sequence. The canonical form of a key (used for tracing below) is an open brace followed by a comma separated listing of the key's atributes in lexicographic order, followed by a close brace. A single space should follow each comma. Multiple keys are listed in lexicographic order.

Program tracing Tracing output should follow the scheme illustrated by the following example. Relation signatures, dependencies, and keys should be in canonical form.

```
#reln: R(A, B, C, D, E);
#deps: A, B -> C, D, E; B, C, D -> A, E; D -> E;
#keys: {A, B}; {B, C, D};
#viol: D -> E;
#deco: R; D -> E;
#rslt: R1(D, E); R2(A, B, C, D);
#reln: R1(D, E);
#deps: D -> E;
#keys: {D};
#viol:
#reln: R2(A, B, C, D);
#keys: A, B -> C, D; B, C, D -> A;
#keys: {A, B}; {B, C, D};
#viol:
```