The main task in this assignment is implementing a program, *calctrans*, that maps calculus queries to algebra queries, using the method described in the textbook.[1] The program calctrans should read its input, which is a sequence of calculus queries, from the standard input and write its output, a similar sequence of algebra queries that are the translations of the corresponding input queries, to the standard output. The output is interpreted as a relational algebra program. Since each calculus query is delimited using braces, the queries in the input do not need any special separators. For the output, we will use the convention of assigning the name `Q_`i to the expression that is equivalent to the i'th query. If the i'th input query is not *safe range* then `Q_`i should be assigned the special token `.z` in the output. If there is any other error in the i'th input query then `Q_`i should be assigned the token `.Z`. All other queries, i.e., all safe range queries, should be correctly translated to equivalent algebra queries.

Both input and output use the syntactical conventions of the textbook except for the lexical simplifications outlined below. For convenience, the special lexical tokens we introduce all have a simple format: a period followed by a single character. In addition to the tokens `.z` and `.Z` introduced earlier, the input uses the following tokens for the indicated purposes in calculus queries. A similar set of tokens for algebra queries is introduced later.

$\wedge$	$\vee$	$\neg$	$\rightarrow$	$\leftrightarrow$	$\exists$	$\forall$
`.a`	`.o`	`.n`	`.i`	`.e`	`.E`	`.A`

For example, the query from Example 5.3.1 of the textbook,

$$\{x_t \mid \exists x_a \text{Movies}(x_t, \text{“Hitchcock”}, x_a) \wedge \neg\text{Movies}(x_t, \text{“Hitchcock”}, \text{“Hitchcock”})\}$$

is represented in the input as

```
{ x_t | .E x_a Movies(x_t, "Hitchcock", x_a)
      .a .n Movies(x_t, "Hitchcock", "Hitchcock") }
```

The second query from that example,

$$\begin{aligned}\{x_t \quad \mid \quad & \exists x_d, x_a \text{Movies}(x_t, x_d, x_a) \\ & \wedge\forall y_a(\exists y_d \text{Movies}(x_t, y_d, y_a) \rightarrow \exists z_t \text{Movies}(z_t, \text{“Hitchcock”}, y_a))\}\end{aligned}$$

is represented as

```
{ x_t | .E x_d, x_a Movies(x_t, x_d, x_a)
      .a .A y_a ( .E y_d Movies(x_t, y_d, y_a)
            .i .E z_t Movies(z_t, "Hitchcock", y_a) ) }
```

[1]Serge Abiteboul, Richard Hull, and Victor Vianu, *Foundations of Databases* (Addison-Wesley, 1995), Section 5.4.

The output should use the following tokens for the indicated purposes:

π	σ	$\bowtie$	δ	$\cup$	$-$	$\rightarrow$	α_β	**diff**
`.p`	`.s`	`.j`	`.r`	`.u`	`.d`	`.t`	α'`[`β'`]`	`.D`

The subscripting suggested by the penultimate column applies only when the subscripting is part of an operator, such as selection or projection, and not, for instance, when subscripts are part of variable names. Recall that the **diff** operator is defined on page 89 of the textbook.

Using the above notation, the following relational algebra program, from Example 5.4.9 of the textbook,

$$\begin{array}{rcl} E_1 & := & (\delta_{A_1A_2\rightarrow xy}(R) \bowtie \{\langle w : b\rangle\}); \\ E_2 & := & (\sigma_{v=x}(E_1 \bowtie \delta_{x\rightarrow v}(E_1)))\ \textbf{diff}\ \delta_{C_1C_2\rightarrow vw}(T); \end{array}$$

is expressed as

```
E_1 := (.r[A_1 A_2 .t x y](R) .j {<w :  b>};
E_2 := (.s[v = x](E_1 .j .r[x .t v](E_1))) .D .r[C_1 C_2 .t v w](T);
```

You should test your implementations on a variety of calculus queries but, at the very least, you should test them on the two queries in Example 5.4.9 of the textbook. Your submission should include a plain-text input file `eg1-in` that contains those two queries in the above input representation. Invoking `calctrans < eg1-in` should produce a relational algebra program in which the expressions assigned to `Q_1` and `Q_2` are the translations of the two input queries. If the output produced by your program is not identical to the result of expressing the algebra programs in that example using our notation, you should explain the discrepancy in a README file that is part of your submission.

Packaging and Submission: Please follow the packaging and submission procedure described in the first assignment, making the obvious change of replacing `hw01` with `hw03`. Further, invoking `make` as described there should produce the executable file `calctrans` noted earlier. Remember to include a README file and a Makefile with the usual contents. If your implementation uses pure Java, you may assume JRE version 1.5. (That is, you are not limited by the older JRE on Gandalf.)