FM is a small language designed for expressing the content of flip movies. It uses objects of externally defined types, allows calls to methods on those objects, and understands simple expressions and the "if" control statement.

Note that the productions have been re-factored since the last assignment (e.g. 4 became 4.1 and 4.2) in order to remove left recursion and simplify your implementation.

1. *program* → **movie** *name* **{** *movieBody* **}** **EOF**
2. *movieBody* → *prologBlock pageBlocks* | *pageBlocks*
3. *prologBlock* → **prolog {** *prologStatements* **}**

4.1 *prologStatements* → *prologStatement prologTail*

4.2 *prologTail* → *prologStatement prologTail* | ε

5. *prologStatement* → *variableDeclaration*

11. *variableDeclaration* → *id* **:** *type***();** | *id* **:** *type***(***exprList***);**

12.1 *pageBlocks* → *pageBlock pageBlocksTail*

12.2 *pageBlocksTail* → *pageBlock pageBlocksTail* | ε

13. *pageBlock* → **show (** *integer* **) {** *pageStatements* **}**

14.1 *pageStatements* → *pageStatement pageTail*

14.2 *pageTail* → *pageStatement pageTail* | ε

15. *pageStatement* →
    { *pageStatements* **}**
    | *methodCall;*
    | *id* = *expr***;**
    | **if (***boolExpr***)** *pageStatement*
    | **if (***boolExpr***)** *pageStatement* **else** *pageStatement*

16.1 *expr* → *term exprTail*

16.2 *exprTail* → + *term exprTail* | - *term exprTail* | ε

17.1 *term* → *factor termTail*

17.2 *termTail* → * *factor termTail* | / factor termTail | ε

18. *factor* → *integer* | *real* | **(** *expr* **)** | *id* | *methodCall*

19.1 *methodCall* → *id callEnd*

19.2 *callEnd* → () | (*exprList*) | .id() | .id(*exprList*)

20.1 *exprList* → *expr exprListTail*

20.2 *exprListTail* → **,** *expr exprListTail* | ε

21. *boolExpr* → *relExpr* | **!** (*relExpr*)

22. *relExpr* → *expr* == *expr* | *expr* > *expr* | *expr* < *expr*

23. *type* → *id*

Language Notes

Comments, blanks, and other whitespace are ignored except as needed to separate adjacent syntactic tokens. A comment begins with the token // and continues to the end of the line.

There are three undefined nonterminals in the grammar: *id*, *integer*, and *real*. An *integer* consists of 1 or more digits (0-9) and denotes a decimal integer. A *real* consists of one or more digits (0-9) followed by a decimal point ".", followed by one or more digits (0-9). An identifier *id* must begin with a letter, and consists of 1 or more letters, digits, and underscores. Upper- and lower-case letters are distinct, thus aa, AA, Aa, and aA are four different identifiers.

The keywords in the grammar (**movie**, **if**, etc.) are reserved and may not be used as identifiers.

All integer values are 32-bit, two's complement numbers.

The fm language includes binary arithmetic operators +, -, * and /. There are no unary + or - operators. The value -n can be computed by evaluating 0-n.

A *bool-exp* is a logical expression, which may only be used as a condition in an if statement. Logical expressions do not have integer values and cannot be stored in variables.

In conditional statements, each else is paired with the nearest previous unpaired if.