Formal Languages and Compilers

Laboratory n° 4

1 Exercise

Write, using JFLEX and CUP, a parser which recognizes the language described below. The program must be able to indicate the wrong structures (rules, facts and interrogations) using the predefined symbol **error**.

1.1 Input Language

A logic program consists of a non-empty set of **facts**, an (eventually) empty set of **rules**, a single **interrogation** and an arbitrary number of **comments**. All sets can appear in any order.

A fact consists of a **predicate** followed by the character '.'.

A rule is composed of a predicate followed by the symbol ':-' followed by a non-empty list of predicates separated by the character ',' and terminated by the character '.'.

An interrogation consists of the symbol '?-' followed by a non-empty list of predicates separated by the character ',' and terminated by the character '.'.

A comment is a string of characters within the symbols '/*' and '*/'.

A predicate is composed of a **functor** followed by a non-empty list of arguments separated by the character ',' terminated by the character ')'; alternatively a predicate is simply an **atom**,

A functor is an atom immediately followed by the character '('.

An argument is a predicate or a **variable**.

An atom is a string of letters, numbers and '_' whose first character is a lowercase letter or a real or integer number, with or without exponent, with or without sign.

A variable is a string of letters, numbers and '_' whose first character is an uppercase letter or the character '_'.

The program must indicate row and column where an error occurred.

1.2 Input file example

```
/* Logic program example */
/* list input */
member(X,cons(X,_)).
member(X,cons(_,Y)):-
member(X,Y).
/* starting list */
start_list(cons(a,cons(b,cons(c,nil)))).
/* interrogation */
?- start_list(L), member(X,L), goal(X).
/* goal */
goal(c).
```

2 Exercise (mini C - Syntax Error Handling)

Starting from the scanner and parser for the **mini C** language written in the previous laboratory, use the predefined symbol **error** to handle language syntax errors.

For example, the parser will report the following syntax errors, showing in the case of wrong input file the row and column where the error occurred:

- *Error in declaration*: variable declaration error
- *Missing ; before* }: missing ';' symbol after a statement
- *Error in expression*: mathematical, boolean or comparison expression error

- *Error in assignment*: assignment error
- *Error in 'print' instruction*: `print` instruction error
- *Error 'else' expected in 'if' instruction*: the keyword `else` is missing in a `if` construct
- *Error in 'if' condition*: an error in the condition of an `if` construct
- *Error '(' expected in 'if' instruction* or *Error ')' expected in 'if' instruction*: if a symbol '(' or ')' misses in a `if` instruction
- *Error in 'while' condition*: an error within the `while` construct condition
- *Error '(' expected in 'while' instruction* o *Error ')' expected in 'while' instruction*: if a symbol '(' or ')' misses in a `while` instruction
- *Error in vector*: error accessing a vector, e.g. missing '[' or wrong symbol or symbols sequence within the square brackets used for vector element access
- *Error in statement*: generic statement error