Domanda 4. (6 punti)

Considera il seguente frammento dell'implementazione del parser per Fasto incluso nel file `Parser.grm` del progetto sviluppato in laboratorio:

```
%token <(int*int)> PLUS TIMES LET IN EQ
%token <string*(int*int)> ID

%nonassoc letprec
%left OR
%left AND
%left DEQ LTH
%left PLUS MINUS

%type <Fasto.UnknownTypes.Exp> Exp

Exp :   ...
      | LET ID EQ Exp IN Exp %prec letprec
          { Let (Dec (#1 $2, $4, $3), $6, $1) }
```

(a) Spiega l'uso di `%prec letprec`

(b) Spiega la regola corrispondente all'espressione `let`.

Soluzione

[dal manuale di mosml]

Parte (a)

The section declares the precedence and associativity of the given symbols. The symbols are usually tokens, but can also be dummy nonterminals (such as `letprec`), for use with the `%prec` directive inside the rule. The directive '`%prec` symbol? may occur among the symbols in a rule right-hand side, to specify that **the rule has the same precedence and associativity as the given symbol**.

Recall that all symbols on the same line are given the same precedence. They have higher precedence than symbols declared in previous `%left, %right, %nonassoc` lines. They have lower precedence than symbols declared in subsequent `%left, %right, %nonassoc` lines. The sym- bols are declared to associate to the left (`%left`), to the right (`%right`), or to be non-associative (`%nonassoc`).

Parte (b)

Semantic actions are arbitrary Moscow ML expressions, which are evaluated to produce the semantic attribute attached to the defined nonterminal. The semantic actions can access the semantic attributes of the symbols in the right-hand side of the rule with the $ notation: $1 is the attribute of the first (leftmost) symbol, \$2 is the attribute of the second symbol, etc. An empty semantic action evaluates to () : unit. Actions occurring in the middle of rules are not supported. Error recovery is not implemented.

Spiegazione:

#1 \$2 significa: prendi il primo elemento della tupla che rappresenta il valore di \$2 = id, cioe' la tupla $< string * (int * int) >$, e quindi la stringa che corrisponde al valore di id. Gli altri sono ovvi. Nota che \$1 alla fine esprime la posizione del comando let nel programma. Infatti il suo valore è di tipo $< (int * int) >$.