LPL grammar (*N** denotes 0, 1 or more repetitions of *N*)

Program	→ *FunDecl FunDecl**
FunDecl	→ *id* **(** *IdList* **) {** *Stm** **}**
IdList	→ *id IdNext**
	→
IdNext	→ **,** *id*
Stm	→ *id* **=** *Exp* **;**
	→ **return** *Exp* **;**
	→ **if (** *Exp* **) {** *Stm** **} else {** *Stm** **}**
	→ **printint** *Exp* **;**
	→ **printchar** *Exp* **;**
	→ *id* **(** *ExpList* **) ;**
Exp	→ *SimpleExp op SimpleExp*
	→ *SimpleExp*
SimpleExp	→ *INTEGER_LITERAL*
	→ *id*
	→ *id* **(** *ExpList* **)**
	→ **(** *Exp* **)**
ExpList	→ *Exp ExpNext**
	→
ExpNext	→ **,** *Exp*

op is one of the following five binary operators: **< == + - ***

id is a sequence of letters, digits and underscores, starting with a letter.

INTEGER_LITERAL is a non-empty sequence of decimal digits. [Note that this means that negative numbers are *not* integer literals.]

Comments: these can either be placed between `/*` and `*/` or make up the remainder of a line beginning with `//`