SPL_PROG → 'glob' '{' VARIABLES '}'
'proc' '{' PROCDEFS '}'
'func' '{' FUNCDEFS '}'
'main' '{' MAINPROG '}'

VARIABLES → VAR VARIABLES | ε
VAR → id

PROCDEFS → PDEF PROCDEFS | ε
PDEF → id '(' PARAM ')' '{' BODY '}'

FUNCDEFS → FDEF FUNCDEFS | ε
FDEF → id '(' PARAM ')' '{' BODY ';' 'return' ATOM '}'

BODY → 'local' '{' MAX3 '}' ALGO
PARAM → MAX3

≤3 names, factored for LL(1)
MAX3 → ε | id MAX3_2
MAX3_2 → ε | id MAX3_1
MAX3_1 → ε | id

MAINPROG → 'var' '{' VARIABLES '}' ALGO

ATOM → id | number

one-or-more INSTR separated by semicolons
ALGO → INSTR ALGO'
ALGO' → ';' INSTR ALGO' | ε

```
INSTR     → 'halt'
          | 'print' OUTPUT
          | id INSTR_AFTER_ID
          | LOOP
          | BRANCH

# disambiguates "call" vs "assignment" after an id
INSTR_AFTER_ID → '(' INPUT ')'        # procedure call
               → '=' ASSIGN_RHS       # assignment

# RHS of assignment; id-case is factored for LL(1)
ASSIGN_RHS     → id ASSIGN_RHS_ID'
               → number
               → PARENS_TERM

ASSIGN_RHS_ID' → '(' INPUT ')' | ε     # function call or just the id atom

PARENS_TERM    → '(' UNOP TERM ')'
               → '(' TERM BINOP TERM ')'

TERM      → ATOM | PARENS_TERM

UNOP      → 'neg' | 'not'
BINOP     → 'eq' | '>' | 'or' | 'and' | 'plus' | 'minus' | 'mult' | 'div'

OUTPUT    → ATOM | string             # the sheet writes “OUTPUT := string”
                                      # we normalise it as an alternative
INPUT     → ε | ATOM INPUT1
```

INPUT1 → ε | ATOM INPUT2

INPUT2 → ε | ATOM