Let's define a language... OCaml-lite!

KEY:
- token
- variable
- metasyn

```
prgm ::= defn ;; defn ;; ... defn ;; exec

defn ::= let name params = stmt
       | let rec name params = expn

params ::= ( name , ... , name )

stmt ::= let name = expn in stmts
       | let name = ref expn in stmts
       | name := expn
       | if expn then blck
       | if expn then blck else blck
       | while expn do stmts done
       | name ( expn , ... , expn )

blck ::= begin stmts end

stmts ::= stmt ; stmt ; ... ; stmt ;

exec ::= let _ = stmt
```

predefined: print_string

The language (cont'd)

```
expn ::= expn + expn | (expn)
       | expn < expn | expn = expn
       | expn && expn
       | name | !name
       | - expn | not expn
       | name (expn, ..., expn)
       | if expn then expn else expn
       | let name = expn in expn
       | 42 | true | "truth" | ()

type ::= int | bool | string | unit

binop ::= + | - | * | / | mod
        | < | <= | > | >= | = | <>
        | && | ||
        | ^
```

predefined: read_line, string_of_int, int_of_string