1. Classic expression grammar

```
Expr ::=  Expr + Term    |    Expr - Term    | Term
Term ::=  Term * Factor  |    Term / Factor  | Factor
Factor ::= ( Expr )      |    num            | ident
```

2. Expression grammar, all ops right associative, prec(()) > prec(@) > prec(#) > prec(%)

E ::= F **%** E | F
F ::= G # F | G
G ::= H **@** G | H
H ::= **(** E **)** | **num** | **ident**

3. Simpler expression grammar -- loses precedence and associativity (not an unambiguous grammar)

```
E ::=     E + E     |    E - E     |    E * E     |
          E / E     |    ( E )     |    num       | id
```

4. Classic expression grammar, immediate left recursion removed

```
E    ::=  T E′
E′   ::=  + T E′     |    - T E′     | ε
T    ::=  F T′
T′   ::=  * F T′     |    / F T′     | ε
F    ::= ( E )       |    num        | id
```

5. Expression grammar, unary (~, ()), binary (+, =), ternary (?:) operators

E ::= **id** = E | F
F ::= F **?** G **:** G | G
G ::= G + H | H
H ::= **(** E **)** | ~ H | **id** | **num**