First set:

	Expr	Expr'	Term	Term'	Factor
First	(, id, num	+, -, ε	(, id, num	*, /, ε	(, id, num

Follow set:

Expr	Expr'	Term	Term'	Factor
eof,)	eof,)	+, -, eof,)	+, -, eof,)	*, /, +, -, eof,)

Algorithm:

1. For each production A → α of the grammar, perform steps 2, 3 and 4.
2. For each terminal a in First(a), add the production A → α to M[A, a]
3. If ε is in First(α), add A → ε to M[A, b] for each terminal b in Follow(A).
4. If ε is in First(α) and **eof** is in Follow(A), add A → α to M[A, **eof**].
5. Make each undefined entry be an **error**.

Grammar:

```
Expr      → Term Expr'
Expr'     → + Term Expr' | - Term Expr' | ε
Term      → Factor Term'
Term'     → * Factor Term' | / Factor Term' | ε
Factor    → (Expr) | num  | id
```

Parse table:

	eof	+	-	*	/	(	)	id	num
Expr									
Expr'									
Term									
Term'									
Factor									