Lab 7: Parsable grammars

Build a grammar (not regular) for the language of arithmetic expressions built from

- Single-digit integers (e.g., 0,1,2,...,8,9)
- The operators `+`, `-`, `*`, `/` and `^`, where `a^b` means a^b.
- Parenthesized sub-expressions.

All of your operators should be parsed with the proper precedence (`^` has the highest precedence, followed by `*`, `/` and then `+`, `-`) and associativity (`+`, `-`, `*`, `/` are left-associative, `^` is right-associative). You may choose to make your grammar prefer left-recursion, right-recursion, or some mixture. Your grammar does not need to conform to any particular normal form. Your grammar should, however, be unambiguous.

To "test" your grammar, build derivation trees for the following expressions, to verify that they are parsed correctly (with respect to precedence and associativity). Since your grammar is unambiguous, there should be only one parse tree for each expression.

```
1 + 2 + 3
```

```
1 ^ 2 ^ 3
```

```
1 + 2 * 3
```

```
(1 + 2) * 3
```