Exercise 2: Grammar of Simplified Boolean Expressions

Write a grammar for Boolean expressions that has identifiers (ident) as well as the constants true and false as operands and &&, || and ! as operators (where ! takes precedence over &&, and && takes precedence over ||). It should also be possible to enclose subexpressions in parentheses. Use the grammar of arithmetic expressions from Chapter 1.4 as a template. Example:

```
(big || small) && ready || big && ! ready
```

Solution

As with arithmetic expressions, we need a top-level production for expressions build with the operator ||, a lower-level production for subexpression built with the operator &&, and a bottom-level production for the operands and the negation (!). This results in the following grammar:

```
BoolExpr = BoolTerm {"||" BoolTerm}.
BoolTerm = BoolFactor {"&&" BoolFactor}.
BoolFactor =
    ident
|   "true"
|   "false"
|   "!" BoolFactor
|   "(" BoolExpr ")".
```

The order of the productions also defines the priority of the operators. The operator ! is at the lowest level in the grammar and therefore takes precedence over the operator &&, which is at the middle level, and this in turn takes precedence over the operator ||, which is at the highest level of the grammar. The expression

```
(big || small) && ready || big && ! ready
```

is recognized as shown in the following syntax tree