HW 8 **Prof. Caldwell**
COSC 3015

1 Integers

The positive integers have representations as sequences of 1 or more digits. A negative integer has the form “$(-k)$” where k is a positive integer.

Exercise 1.1. Write a parser `intp` that parses integers of this form. You can use the basic parsers provided in the file Parser.hs. Here is some example behavior:

```
*Expr> :t intp
intp :: Parser Int
*Expr> parse intp "10"
[(10,"")]
*Expr> parse intp "01"
[(1,"")]
*Expr> parse intp "(-10)"
[(-10,"")]
*Expr> parse intp "(-10 "
[]
*Expr> parse intp " -10 "
[]
*Expr> parse intp "0000"
[(0,"")]
```

2 Expressions

Consider the following concrete grammar for expressions.

expr ::= *term* (’+’ *expr* | ϵ)
term ::= *factor* (’*’ *expr* | ϵ)
factor ::= ’(’ *expr* ’)’ | *nat*

Here is a parser that calculates the values of expressions in this language.

```
expr :: Parser Int
expr = do t <- term
          do symbol "+"
             e <- expr
             return (t + e)
```

```
            +++ return t

term :: Parser Int
term = do f <- factor
          do symbol "*"
             t <- term
             return (f * t)
            +++ return f

factor :: Parser Int
factor = do symbol "("
            e <- expr
            symbol ")"
            return e
           +++ natural
```

Now, consider the following expression datatype.

```
data BinOp = Add | Times
   deriving Show

data Exp = Const Int | BinExp BinOp Exp Exp
   deriving Show
```

Exercise 2.1. Modify the parsers `expr`, `term`, and `factor` of type `Parser Int` into parsers that build instances of the `Expr` type, *i.e.* the new parsers should have type `Parser Expr`.

Here are some example runs:

```
*Expression> parse expr' "55"
parse expr' "55"
[(Const 55,"")]
*Expression> parse expr' "55 + 23"
parse expr' "55 + 23"
[(BinExp Add (Const 55) (Const 23),"")]
*Expression> parse expr' "55 + 23 + 24"
parse expr' "55 + 23 + 24"
```

```
[(BinExp Add (Const 55) (BinExp Add (Const 23) (Const 24)),"")]
*Expression> parse expr' "55 + 23 + 24 * 25"
parse expr' "55 + 23 + 24 * 25"
[(BinExp Add (Const 55) (BinExp Add (Const 23)
 (BinExp Times (Const 24) (Const 25))),"")]
*Expression> parse expr' "55 + (23 + 24) * 25"
parse expr' "55 + (23 + 24) * 25"
[(BinExp Add (Const 55) (BinExp Times
             (BinExp Add (Const 23) (Const 24)) (Const 25)),"")]
*Expression>
```