HW 20 **Prof. Caldwell**
Due: 16 November 2010 **COSC 3015**

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 0.1. Write a parser `intp` that parses integers of this form. Here is some example behavior:

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

Now, consider a language of expressions of the following form.

```
data Expr  = Const Int | Add Expr Expr
```

Exercise 0.2. Write a parser `expr` of type `Parser Expr` for parsing strings containing additions of integers into the `Expr` type. (You will want to use your `intp` parser from exercise 1.) So for example:

```
*Expr> apply expr "10"
[(C 10,"")]
*Expr> apply expr "10 + 12"
[(Add (C 10) (C 12),"")]
*Expr> apply expr "10 + (-12)"
[(Add (C 10) (C (-12)),"")]
*Expr> apply expr "10 + (-12) + 14"
[(Add (C 10) (Add (C (-12)) (C 14)),"")]
```