HW 20 **Prof. Caldwell**
Due: 4 December 2008 **COSC 3015**

1 Parsing terms

In class we presented code to parse Types. Inthis assignement you will write code to parse terms and to build elements of th data type `Term`.

```
data Term = V String
          | Ap Term Term
          | Abs String Term
          | Spread Term (String,String) Term
          | Pair Term Term
 deriving Eq
```

The BNF form for the terms is given as follows:

$$
\begin{array}{rcl}
term & ::= & id \\
 & & \mid \text{"("}\ term\ term\ \text{")"} \\
 & & \mid \text{"\textbackslash"}\ id\ \text{"."}\ term \\
 & & \mid \text{"}\texttt{spread}\text{"}\ \text{"("}\ term\ \text{";"}\ id\ \text{","}\ id\ \text{"."}\ term\ \text{")"} \\
 & & \mid \text{"<"}\ term\ \text{","}\ term\ \text{">"}
\end{array}
$$

Here, `id` is the an identifier, parsed with the parser `identifier` (included int he code.

1.1 An interactive typechecker

I have written a small interpreter (used in the file Main.hs) allowing you to enter a term and then it prints the type of the term, if any. For example:

```
*Main> typecheck

[("y",a)] |- (\x->y) :: (b -> a)
enter_term: \x.\y.\z. ((x y) (y z))

[] |- (\x->(\y->(\z->((x y) (y z))))) :: (((a -> b) -> (b -> c)) -> ((a -> b) -> (a -> c)))
```

Exercise 1.1. Write the parsers `ApP` (for apply terms), `absP` (for abstraction terms), `spreadP` (for spread terms) and `pairP` (for pair terms) and put them in the file `Parser.hs`. Load `Main.hs` and run the interpreter by evaluating `typechecker`. Extra credit will be given if you run the typechecker on some especially interesting terms.