CS2104 A Parser for Lambda Calculus
Lab/Assignment 4. (Deadline 12 November 2007 Mon, 6PM)

Note : Submit in a single file, say lab4.oz. There is no need to provide any test cases in your submission.

In the previous assignment, we introduced a small language based in lambda calculus that is denoted by the following Oz data type.

```
<Expr>  ::=  <Id> | lam(<Id> <Expr>) | apply(<Expr> <Expr>)
                  | let(<Id>#<Expr> <Expr>)
```

Consider a textual version of the above language with the following grammar rules:

```
<Alpha>    ::=  [a-zA-Z]
<AlphaNum> ::=  [a-zA-Z0-9]
<Id>       ::=  <Alpha><AlphaNum>*
<Expr>     ::=  <Id>   | lambda <Id>+ . <Expr> end
                       | <Expr> <Expr>
                       | ( <Expr> )
                       | let <Id>=<Expr> in <Expr> end
```

Write a parser for this language that could convert each expression (in string format) into its corresponding abstract syntax tree (as above Oz data structure). Some examples of this conversion are:

```
lambda x y1 . y1 x end
              converts to: lam(x lam(y1 apply(y1 x)))
x y z
              converts to: apply(apply(x y) z)
x (y z)
              converts to: apply(x apply(y z))
let x=y in x
              converts to: an exception/error!
```

You are provided a module, lexer.ozf, exporting a lexer function which processes an input string by looking at its characters and categorizing them into the following tokens:

- tkEq – token equal
- tkLBk – token left bracket
- tkRBk – token right bracket
- tkDot – token dot
- tkKwLambda – token keyword lambda
- tkKwLet – token keyword let
- tkKwIn – token keyword in
- tkKwEnd – token keyword end
- tkId(list of Ascii codes corresponding to the identifier's characters) – token identifier
- tkEOF – token end of file

You may start the assignment using the code from lab4.oz. This file shows how to invoke the lexer function from the lexer.ozf module. The module is available on the course webpage in two formats: compiled format (ozf file) and source format (oz file), but you will need only the ozf file. The source of the library (lexer.oz) is provided only for reference.

Usage example (from lab4.oz):

```
{Browse {Lexer.lexer "let x1=y in x1 end"}}
% produces [tkKwLEt tkId([120 49]) tkEq tkId([121]) tkKwIn tkId([120 49]) tkKwEnd tkEOF]

{Browse {Lexer.lexer "lambda x y1 . y1 x end"}}
% produces [tkKwLambda tkId([120]) tkId([121 49]) tkDot tkId([121 49]) tkId([120]) tkKwEnd tkEOF]
```

In the above examples, 120 = Ascii code for x, 49 = Ascii code for 1, 121 = Ascii code for y.