COP4342 - 2006 Fall

Assignment 6

Using emacs lisp, bison, and flex

Objectives: Learn how to write simple code in emacs lisp, bison, and flex.

Instructions: Your assignment is to write two parsers, one in bison and flex, and the other in emacs lisp.

The BNF grammar for the "language" to be parsed is:

```
<program> ::=  "program" <id> <variablesSection> <functionsSection> <statementsSection> "end"

<variablesSection> ::= "variables" "{" <variableDeclaration>* "}"

<variableDeclaration> ::= <id> ":" <id> ";"

<functionsSection> ::= "functions" "{" <functionDeclaration>* "}"

<functionDeclaration> ::= "define" <id> ":" <id> "(" [<argsList>] ")" "{" <statement>* "}"

<argsList> ::= <argPair> | <argsList> "," <argPair>

<argPair> ::= <id> <id>

<statementsSection> ::= "statements" "{" <statement>* "}"

<statement> ::= "var" <variableDeclaration> | <whileLoop> | <ifStruct> | <subroutineCall> ";"

<whileLoop> ::= "while" "(" <subroutineCall> ")" "{" <statement>* "}"

<ifStruct> ::= "if" "(" <subroutineCall> ")" "{" <statement>* "}" [ "else" "{" <statement>* "} ]

<subroutineCall> ::= <id> "(" [<callArgsList>] ")"

<callArgsList> ::= <id> | <callArgsList> "," <id>
```

The program should send out "okay" if it successfully parses a file or "not okay" if it finds an error.

Your parsers should be able to recognize all of the files located off the class web page labelled "Verify Simple Parser", and should fail on all of the files located off the class web page labelled "Check Simple Parser".

Bison/Lex: Create a Bison source file called **assign6.y** where you will put the main program and all of the Bison parser production rules. Create a Flex source file called **assign6.l** where you will put the lexical analyzer rules.

If your parser finds a syntax error, report what line was being parsed when the syntax error was found. If your parse succeeds, print an "okay" message which also reports how many lines were found in the file. For example, if the parse succeeds, you should give a message like:

```
file is okay, 101 lines parsed
```

If the parse fails, you should give a message like:

```
file is not okay, syntax error found at line 91
```

Emacs: Create an emacs lisp source file called **assign6.emacs**. Your parser should be a recursive descent parser, as discussed in class, since that is the easiest type of parser to write in Lisp. The name of the parser function should be `simple_parser` and it should be interactive. As part of that interactivity, you should pull the file in every invocation of `simple_parser` with the `find-file-other-window` function, and make sure that your parse always starts at the beginning of that buffer.

If the parse succeeds, use the `message` function to report "file is okay". Leave the character pointer at the end of the file.

If the parse fails, use the `beep` function and the `message` function to report "file is not okay – problem is here" and leave the character pointer at the point where the error occurred.

These predefined emacs lisp functions should be considered when you are writing your parser, since this program can be completely written using just them as a base for you to write your accompanying lisp functions:

```
and
beep
concat
defun
find-file-other-window
goto-char
interactive
looking-at
match-end
message
or
point-min
```