

CMPS 258 – PROGRAMMING LANGUAGES – SPRING 2021  
WEEK 11 ASSIGNMENT

*This assignment may be done in groups of two.*

In this assignment, you will implement a parser for a small language similar to SML using a parser generator. The parser will generate a tree that represents the code. Typically, the tree will be passed onto a semantic analyzer, but for the purpose of this assignment, the tree will simply be printed to a file.

The grammar you will implement as the following production rules:

```
program ::= binding_list
        | ε
binding_list ::= binding_list binding
            | binding
binding ::= val id = expr
        | fun id id = expr
expr ::= if expr then expr else expr
      | op_expr
op_expr ::= op_expr + op_expr
        | op_expr - op_expr
        | op_expr * op_expr
        | op_expr / op_expr
        | op_expr > op_expr
        | op_expr < op_expr
        | op_expr andalso op_expr
        | op_expr orelse op_expr
        | call_expr
call_expr ::= call_expr basic_expr
          | basic_expr
basic_expr ::= ( expr )
           | id
           | int_const
           | true
           | false
```

All operators are left associative. The order of operator precedence from lowest to highest is:

1. orelse
2. andalso
3. >, <
4. +, -
5. \*, /

The syntax rules for terminals are:

- Keywords (val, fun, andalso, orelse, if, then, else, true, false): are as written
- Identifiers (id): may start with any alphabetical character or underscore, and may contain any number of alphabetical characters, numeric characters, or underscore.
- Integer literals (int\_const): may contain an arbitrary number of numeric characters
- Operators and separators (=, +, -, \*, /, >, <, (, )) are as written
- Whitespace: space, tab (\t), new line (\n), and carriage return (\r) should be ignored

To implement your parser, follow the following steps:

1. Download and install *flex*, a free and open source lexer generator, and *bison*, a free and open source parser generator. On some Linux systems, you can install them using your package manager (e.g., `sudo apt-get install flex`, `sudo apt-get install bison`). On Windows systems, you can obtain them at these links:  
<http://gnuwin32.sourceforge.net/packages/flex.htm>,  
<http://gnuwin32.sourceforge.net/packages/bison.htm>.
2. Edit the file `parser.y` to implement your parser. In the actions for most production rules, you should call the corresponding functions declared in `tree.h` to build the tree. Some examples are provided for you.
3. When you are done editing `parser.y`, use *bison* to generate the parser code by executing the following command in the command line:

```
bison -d parser.y
```

This command will generate a C file called `parser.tab.c` that contains the implementation of your parser. It will also generate a header file called `parser.tab.h` that is included in the corresponding flex file to allow the lexer to pass tokens to the parser.

4. Edit the file `lexer.flex` to implement your lexer. In the actions for most regular expressions, you should pass a token to the parser, and possibly a value associated with the token. Some examples are provided for you.
5. When you are done editing `lexer.flex`, use *flex* to generate the lexer code by executing the following command in the command line:

```
flex lexer.flex
```

This command will generate a C file called `lex.yy.c` that contains the implementation of your lexer.

6. Compile `parser.tab.c` and `lex.yy.c` and `tree.c` using any C compiler of your choice. The generated binary is your parser.
7. Test your program. The generated binary takes the input file name as the first argument (`test.sm1` by default) and the output file name as the second argument (`test.dot` by default). The output file can then be passed to the graph visualization software *dot* to draw the tree (you can also find online *dot* to PDF converters). You are provided with the following reference files to help test your tool: `test.sm1`, `test-reference.dot`, `test-reference.pdf`. Executing your tool with `test.sm1` as input should generate a file identical to `test-reference.dot` which, when drawn, should generate a drawing similar to `test-reference.pdf`.

### Submission Instructions

Submit your modified `parser.y` and `lexer.flex` files via Moodle. Do not submit any other files or compressed folders. Make sure to include a comment in each file with your name and AUBnet ID (e.g., abc01). If you did the assignment in a group of two, only one group member should submit. In this case, make sure to include both group members' names and AUBnet IDs in each file.