Yacc / Bison

Parser generator using Scala-Bison

What is a parser generator

Is a program that takes a grammar and generates a code that can be used to parse the grammar

Why are parser generators used?

Because they generate a state machine and number of the states is not O(n) hence maintaining that code would
be near impossible. Solution is to use parser generator make you concentrate on the grammar instead of its implementation.

Bison vs. Yacc

- Yacc and Bison are closely compatible
- Bison in a part of the GNU project. And yacc is used as a utility on Berkeley Software Distribution (BSD). Though its compatible with yacc, but Lex and Yacc are a thing of the past. Flex and bison are widely used today.

Yacc structure (`*.y`)

- Directives: the first section is where we define tokens, associativity, order of operation and etc
- Rules: in this section we define BNF grammar and it's respective parse yield
- Code: in this section we can share some utility functions or global variable with grammar rule handler code

Notations

- `$$`: this pseudo-variable stands for the semantic value for the grouping that the rule is going to construct
- `$1`, `$2` and etc: semantic values of the components of the rule are referred to as $1, $2, and so on
- `|`: means alternative structure of non-terminal

Example yacc file

```
%{
  #include <stdio.h>
  #define YYSTYPE char const *
  int yylex (void);
  void yyerror (char const *);
%}

%token TYPENAME ID

%right '='
%left '+'

%glr-parser

%%

prog:
  %empty
| prog stmt   { printf ("\n"); }
;

stmt:
  expr ';'  %dprec 1
| decl      %dprec 2
;

expr:
  ID               { printf ("%s ", $$); }
| TYPENAME '(' expr ')'
                   { printf ("%s <cast> ", $1); }
| expr '+' expr    { printf ("+ "); }
| expr '=' expr    { printf ("= "); }
;

decl:
  TYPENAME declarator ';'
                   { printf ("%s <declare> ", $1); }
| TYPENAME declarator '=' expr ';'
                   { printf ("%s <init-declare> ", $1); }
;

declarator:
  ID               { printf ("\"%s\" ", $1); }
| '(' declarator ')'
;
```

Lab assignment

Parse XML

```
<foo />
<foo>&lt;</foo>
<foo><bar />&gt;</foo>
<  foo  />
&amp;
</  foo  >
<foo>  <bar />  </foo>
<foo a='b' c="d" e="f" />
<foo a  =  '"'b ="&lt;&lt;" c  =  "d" />
<!-- ignored --> <foo /> <!-- also ignored -->
<!-- so lonely & <><-><> nobody listens to me -->
<![CDATA[this is so <neat /> I'm not even a <tag />! <![CDATA[ no nesting]]>
   not trimmed
"not a string, &amp; just some text"
<bar string="oh yeah here's a quote ' and an entity &quot;" />
<if> 5>7 <!-- > is allowed loose -->
<then> <do action="something"/> </then>
<else> <do action="something else"/>
</if>
```