McMaster University Comp Sci 4TB3/6TB3, Winter Term 2017/18 — Lab 7
For the Labs on March 6 - March 9,
Due Monday, March 12, 11 pm

Eden Burton, Jenny Wang, Spencer Park
out of 24 points; extra points count towards any other lab

- *Submission is to be done exclusively through Avenue. Submissions via e-mail will* ***not*** *be accepted. A* ***10% penalty*** *will be accessed for each day the lab is submitted after the due date.*
- You need to download the P0 compiler posted together with the lab, it contains some modifications for this lab.
- This assignment requires access to a Linux, MacOS X, or some other Unix computer. You can log in remotely to either `moore.mcmaster.ca` or to `mills.mcmaster.ca` with ssh. Submissions are tested on `moore.mcmaster.ca`, please check if your submission works there.
- In this lab, you are allowed to work in pairs, provided that you split the work equally and arrive at a common understanding of the solution. However, in that case you must state in your submission the person you worked with, such that similarities in the solution will not be construed as Academic Dishonesty. Working in groups of three or larger is not allowed and will be considered Academic Dishonesty. If you look for someone to work with, we will try to find a match, please contact the TAs.
- You are allowed and encouraged to talk to everyone in the course to get a common understanding of the problem, but you can share partial solutions only with your collaborator, if you work in a pair. The final submission must be your own, that is, you cannot submit identical submissions with two names on them.
- In the lab sessions, the solution to last week's lab questions are discussed and you can get help with this week's lab questions. Attendance at the labs is not checked.

Tutorial Exercise 1 (Pretty-Printing)**.** This lab asks you to write a *beautifier* or *pretty-printer* for P0. An example of a program beautifier is *ptop* that comes with *fpc* (see the Free Pascal User's Manual). There are numerous beautifiers available for Java and other languages. Download the version of the P0 compiler for this lab. In addition to generating code, it is supposed to output a pretty-printed version of the program on stdout, with systematic indentation of the control structures and declarations:

- Expressions are assumed to fit on one line and are output without line-breaking, but with one space around all binary operators, one space after unary operators, and no space around parenthesis or brackets.
- Statements, declarations, and types may go over several lines and need to be broken and indented as follow.
- All statements start a new line. Assignments and procedure calls are assumed to fit on one line. The scheme for other statements is:

```
begin       if E then      if E then       while E do
  S           S              S               S
end                        else
                             T
```

That is, we assume that `if E then` and `while E do` fit on one line and we do not break them; keywords `end` and `else` start a new line.

- Each constant, type, variable declaration starts a new line, as in:

```
const          type           var
  c1 = E1;       t1 = T1;       v1: T1;
  c2 = E2;       t2 = T2;       v2: T2;
```

- Type alias declarations are assumed to fit on one line, array and record declarations start a new line; each new field declaration starts a new line:

```
t1 = t2
t1 =
  array [E1 .. E2] of t2
t1 =
  array [E1 .. E2] of
    array [E3 .. E4] of t2
t1 =
  record
    f1: T1
    f2: T2
  end
```

- The program header is not indented, the program declarations and the program body are indented once:

```
program p
  D
  begin
    S
  end
```

- Procedure headers are not indented, local declarations and the procedure body are indented once:

```
procedure p
  D
  begin
    S
  end
```

Define first an attribute grammar for pretty-printing and then implement it by modifying the compiler. The grammar for P0, as extracted from the documentation comment of each parsing procedure, is:

```
    selector = {"." ident | "[" expression "]"}.
    factor = ident selector | integer | "(" expression ")" | "not" factor.
    term = factor {("*" | "div" | "mod" | "and") factor}.
    simpleExpression = ["+" | "-"] term {("+" | "-" | "or") term}.
    expression = simpleExpression
        {("=" | "<>" | "<" | "<=" | ">" | ">=") simpleExpression}.
    compoundStatement = "begin" statement {";" statement} "end"
    statement =
        ident selector ":=" expression |
        ident "(" [expression {"," expression}] ")" |
        compoundStatement |
        "if" expression "then" statement ["else"statement] |
        "while" expression "do" statement.
    type =
        ident |
        "array" "[" expression ".." expression "]" "of" type |
        "record" typedIds {";" typedIds} "end".
    typedIds = ident {"," ident} ":" type.
    declarations =
        {"const" ident "=" expression ";"}
        {"type" ident "=" type ";"}
        {"var" typedIds ";"}
        {"procedure" ident ["(" [["var"] typedIds {";" ["var"] typedIds}] ")"] ";"
            declarations compoundStatement ";"}.
    program = "program" ident ";" declarations compoundStatement.
```

The level of indentation is to be determined by an inherited attribute that is passed to all parsing procedures. Add attribute rules to the grammar for factor, compound statement, procedure call, if-statement, and program and modify the P0 compiler to implement those. For example

```
program p; begin if false then writeln else
if true then write(5) else write(7)
; write(9) end
```

should be pretty-printed as:

```
program p;
  begin
    if false then
      writeln
    else
      if true then
        write(5)
      else
        write(7);
    write(9)
  end
```

Answer. The version of P0 for this assignment contains modifications of the Python functions `factor`, `compoundStatement` and `statement` with the attribute rules in the documentation comments. The example `testPrettyPrint0()` will work, but `testPrettyPrint1()` will fail as required modifications are missing.

Lab Question 1 (Pretty-Printing, 24 points)**.** Extend the attribute rules for pretty-printing the whole language and modify the P0 compiler accordingly. Do so by editing above version of P0. Note that a modification of the compiler without accompanying attribute grammar will get a significant deduction.

Lab Question 2 (Prettier Pretty-Printing, 8 points)**.** Modify the attribute rules and the compiler such that either markdown or html is generated: keywords are to be printed in boldface, identifiers (including standard identifiers like *write*, *integer*, and *true*) in italics. Integer constants and operators should be upright. You may also employ a colour scheme or your choice.

Lab Question 3 (Proper Pretty-Printing, 4 points)**.** So far, pretty-printing will swallow all comments, as these are not passed to the parser and only the parser outputs to stdout. Modify the compiler such that also comments are pretty-printed. It is up to you to decide how to indent them and which font style or colour to use.