McMaster University Comp Sci 4TB3/6TB3, Winter Term 2017/18 — Lab 6
For the Labs on February 13 -February 16,
Due Monday, February 26, 11 pm

Eden Burton, Jenny Wang, Spencer Park
out of 20 points

- *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.*
- 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.
- The Tutorial Exercises will be presented in the tutorials; you need to submit only answers to the Lab Questions. In the labs, 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. There is no specfic tutorial this week. The task is to complete what was started in class.

Lab Question 1 (P0 Scanner, 10 points)**.** Complete the exercise started in class. That is, implement a scanner for the P0 language as described on *pages 128-130* in the course materials. You may use the sample code on Avenue as a starting point, or you may write your own scanner from scratch.

As mentioned before, you can implement the scanner in the language of your choice but you must provide a script called "script." which allows your program to run on a Linux server such as `mills.mcmaster.ca` hosted at the university. The program shall read from a *user specified file* and output the symbols found.

Upon finding a symbol, the program should output text which exactly matches the symbol type constants on page 128, followed by a endline. For example, the trival program below in Listing 1 should yield the described output.

Table 1: Symbol Class to Source Text Mapping

symbol class	input character sequence
TimesSym, DivSym, ModSym,	'*', 'div' , 'mod'
AndSym, OrSym	'and', 'or'
PlusSym, MinusSym	'+', '-'
EqlSym, NeqSym,	'=' , '<>'
LssSym, GtrSym, LeqSym, GeqSym	'<' , '>', '<=', '>='
PeriodSym, CommaSym , ColonSym	'.', ',' , ':'
LparenSym, RparenSym	'(' , ')'
LbrakSym, RbrakSym	'[' , ']'
OfSym	'of'
ThenSym	'then'
DoSym	'do'
NotSym	'not'
BecomesSym	':='
NumberSym	
IdentSym	
SemicolonSym	';'
BeginSym, EndSym	'begin', 'end'
IfSym, ElseSym	'if', 'else'
WhileSym	'while'
ArraySym, RecordSym	'array', 'record'
ConstSym	'const'
TypeSym	'type'
VarSym	'var'
ProcedureSym	'procedure'
ProgramSym	'program'
EofSym	eof

Listing 1: trival p0 program

```
{a sample program}

program sample
  begin

  end
```

Expected Output

```
ProgramSym
IdentSym
BeginSym
EndSym
EofSym
```