SAINT MARY'S UNIVERSITY SINCE 1802 One University. One World. Yours.

Department of Mathematics and Computing Science
CSCI 3430 - Principles of Programming Languages

Assignment #2

Assigned : Monday January 18, 2017
Due : Tuesday January 26, 2017

Hand in source code only. Please add all source code into one big zip file and post that to Moodle, it won't take more than **ONE** file. If it's too large to upload to Moodle, you likely have more than source code. If you are still stuck and can't upload the file, email it to me, cc to yourself and see me in class.

---

1. (3 pt) Regular expressions are great for pattern matching. Write "grep" regular expressions for the following:
   a. All lines exactly 3 characters long
   b. All lines starting with "c", ending with "s" and exactly 5 characters long
   c. All lines that DO NOT contain the word "THAT" (case insensitive)
   d. All lines that contain the word (space delimited) "Here" but not "There" (case sensitive)
   e. All lines that contain "THIS" and "THAT" but not "THE OTHER THING"
   f. An 8 character name, where the first character must be an uppercase letter (not a number), the rest can be letters or numbers, with no internal punctuation, followed by a comma"," followed by the last name. Since first and last name have the same syntax it's a great opportunity to reuse syntax.

You can test these expressions using "grep" or "egrep" or "fgrep" on your favorite UNIX computer and a file containing any of the above patterns or you can use any of the online regex tools. However, the "cs.smu.ca" Ubuntu Linux machine is where I will test it so it MUST work there.

2. (2 pts) Fix up the example grammar included below (shown below as taken from the slides) to include / and * operators with correct precedence and allow an unlimited number of <term>. Make sure the grammar only evaluates one way, grouping / before * and before either + or -.

```
<expr> ::=  <term> + <term> | <term> - <term>
<term> ::=  <var> | const
<var> ::=  a | b | c | d | e
```

3. (2 pts) Draw a parse tree using the rules you created in question 2 for the expression a/b/c+d*e using ASCII Art or some drawing package like Visio, MS PowerPoint, MS Word or even good old MS paint.

4. (3 pts) Write a set of grammar rules that recognizes the following URLs (valid characters, http:// prefix, arbitrary number of "." And "?" and "/" etc). You do not need an exhaustive thing to match ALL possible URLS, just restrict this to a two systems in the following (note that some have trailing slashes and some do not)

   http://www.google.ca
   http://www.google.ca/search?q=yacc
   http://cs.smu.ca/
   http://cs.smu.ca/~curtis
   http://moodle.cs.smu.ca/login/index.php
   http://moodle.cs.smu.ca/course/view.php?id=3
   http://moodle.cs.smu.ca/mod/page/view.php?id=65

This is actually quite easy if you stick to the patterns that are the similar in the above.

BTW: If you want to test the grammar rules, you really can create a very complex regular expression or a complete FLEX program to parse input on the cs computer or your Windows PC, apple or any other device. Note if you use FLEX on your own computer, you will likely have use a virtual machine or download the Gnu YACC or BISON for your own system.