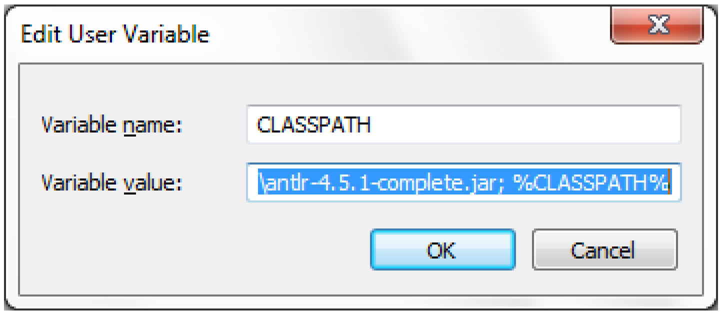
CS330 – Programming Languages
HW4 - ANTLR Expression Parser (Expr.g4)

Name __Joshua Haupt__ Grade __________________

This assignment is designed to get you started experimenting with ANTLR parser generator in windows environment. You will create and experiment with the simple calculator grammar, described in chapter 4 of the ANTLR 4 book. You will then extend the expression grammar to include rules for an ^ operator.

The following is a list of the basic tasks, with some directions/guidelines to help you with the tasks. You can find detailed discussions and explanation in the ANTLR book. In addition, although the grammar is discussed in chapter 4, it is important that you read through the first 3 chapters before starting your assignment.

1. Copy the ANTLR related files from the HW4ANTLR folder on **\\\\vm-01.cs.siue.edu** (also on moodle) and place them in your designated ANTLR directory. Add the complete path of your ANTLR directory to your *CLASSPATH* environment variable. To set environment variable in Windows 7, right-click on Computer (icon or from start menu), select *properties*, then *advanced system setting*, then *advanced*. Then set your CLASSPATH variable,

   Variable name: CLASSPATH
   Variable value: \antlr-4.5.1-complete.jar; %CLASSPATH%

   Where Variable value should be set to (current dir .) and (;) DIR location of ANTLR, ex.,

   .;C:\newD\SIUE\CS330\ANTLR\ANTLR45\antlr-4.5.1-complete.jar; %CLASSPATH%

   ***Note for windows 8/8.1/10 users:**

   To access system properties you must right click the windows icon in the bottom left of your screen, then select system, and then select advanced system settings in the left menu panel. Once you have selected this, a dialog will pop up and there will be a button that says "Environment Variables". Select this button and edit variables as described above.

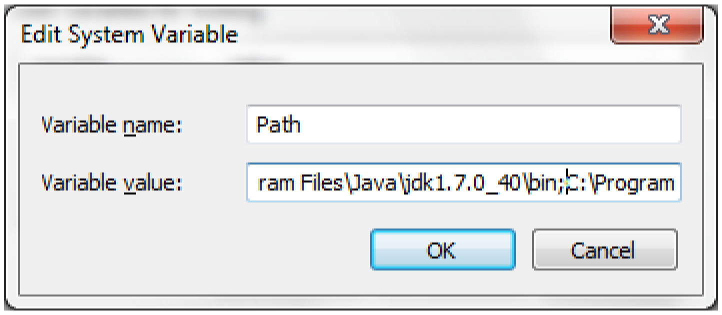
   Use a text editor (ex., nodepad) to create the expression grammar file "Expr.g4" as shown in ANTLR 4 reference book section 4.1. To keep things tidy, you may want to create a separate directory for this and other related files. (**Note,** you could also use Antlrworks 2 (a GUI grammar interface) for creating/editing your g4 files. Additional information about this program including ANTLR example and fresh downloads can be found at **http://www.antlr.org** .)

2. Modify antlr4.bat file to fit your need.

3. Open a DOS command window at your antlr directory by shift-rightCLICK. Run antlr4.bat on Expr.g4 to create java files including ExprParser.java, ExprLexer.java, Expr.token, Notice that antlr4.bat also defines grun.

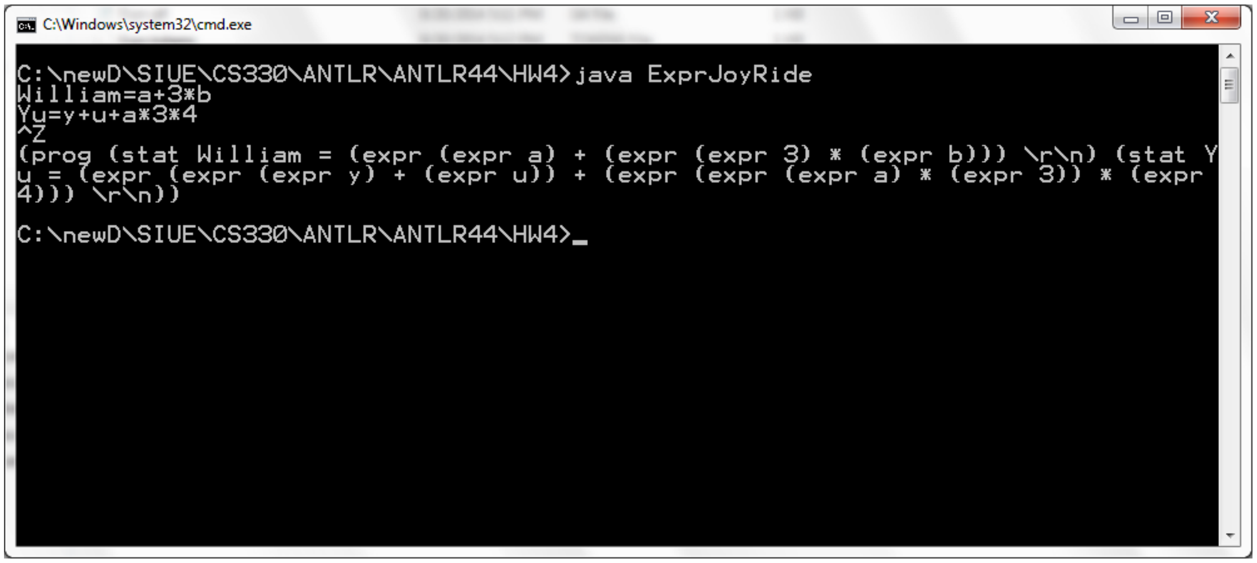
4. Use a text editor to create the driver program named *ExprJoyRide.java* as described in ANTLR 4.1. This file should be in the same directory as the rest of java files, which have already been created with antlr4.

5. Download and install Java Development Kit (5.0 or higher) on your computer, if it's not already installed. Be sure to add the path to your java bin to your PATH system variable, which you can access similar to that of CLASSPATH. For example, on my computer, I added C:\Program File\Java\jdk1.7.0_40\bin as show below. This allows you to run java compiler (javac) anywhere, without having to use the entire path as show below: **C:\Program File\Java\jdk1.7.0_40\bin\javac**



6. Compile all relevant java files to generate the corresponding java class files (javac *.java).

7. Run ExprJoyRide (ex., java ExprJoyRide) at DOS prompt to test it with various expressions. Note that ExprJoyRider does not give prompt, so when it halt, it might just be waiting for your expression. Try enter some simple assignment statement or expression and then ctrl-z to end input. Here is a simple example:



```
C:\newD\SIUE\CS330\ANTLR\ANTLR44\HW4>java ExprJoyRide
William=a+3*b
Yu=y+u+a*3*4
^Z
(prog (stat William = (expr (expr a) + (expr (expr 3) * (expr b))) \r\n) (stat Y
u = (expr (expr (expr y) + (expr u)) + (expr (expr (expr a) * (expr 3)) * (expr
4))) \r\n))

C:\newD\SIUE\CS330\ANTLR\ANTLR44\HW4>_
```

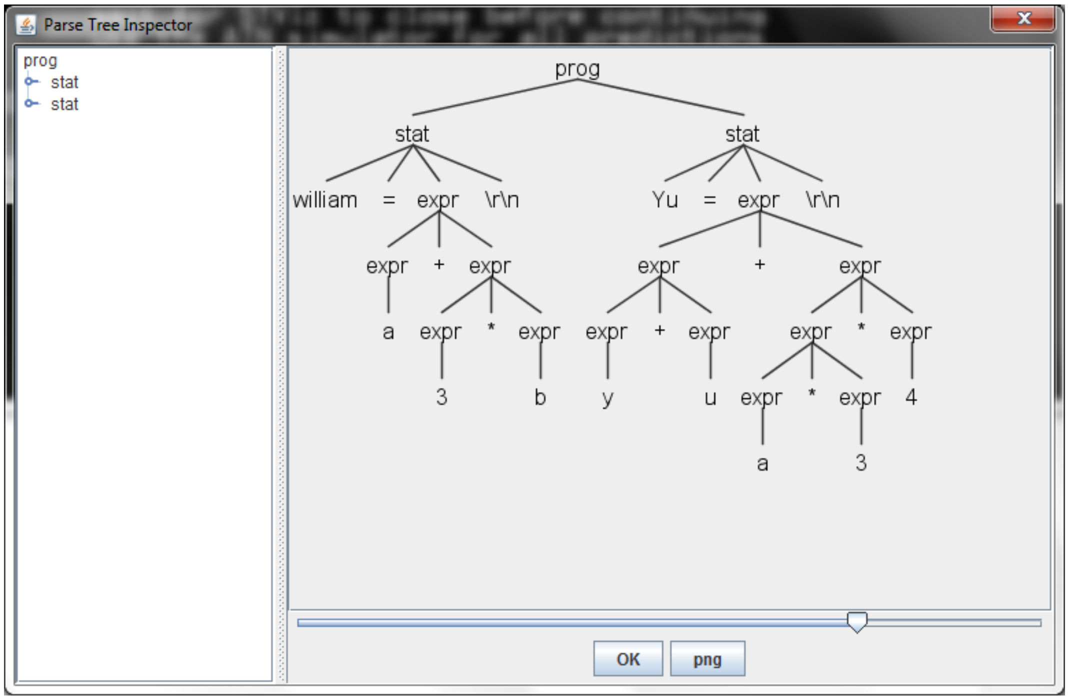
8. Test your grammar using testRig provided by ANTLR4. To run testRig, use the alias grun which is defined in antlr4.bat. You can use –gui to get a gui tree. For example,

   ➢ grun Expr prog –gui

   William=a+3*b

   Yu=y+u+a*3*4

   Create the following gui tree



Note that the results from the two assignment indicates correct operator precedence and associativity.

So far, everything you have done is just following book examples. Now you need to add a little extra to the Expr.g4 grammar.

9. **Extend the Expr grammar to add definition for exponent operator ^.** Note that ^ should be defined as right associative and have precedence higher than * and /. Also note that starting in 4.2, <assoc=right> option must be specified at the beginning of a rule option. For example, if I were to make * right associative, I would use

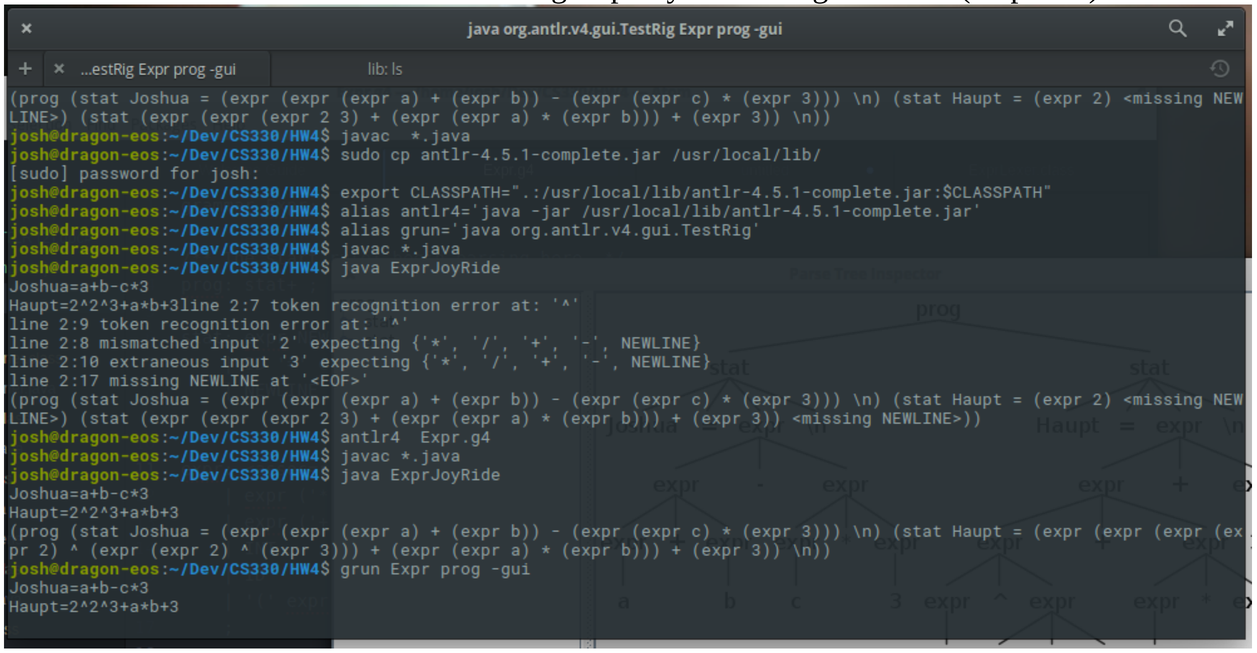
   <assoc=right> expr '*' expr

10. After making changes to Expr.g4, you need to go through steps 4 and 7-9 again to generate and compile java code and test your new grammar with both ExprJoyRide and grun. Use the following input for testing:

    Yourfirstname= a+b-c*3
    Yourlastname= 2^2^3+a*b+3

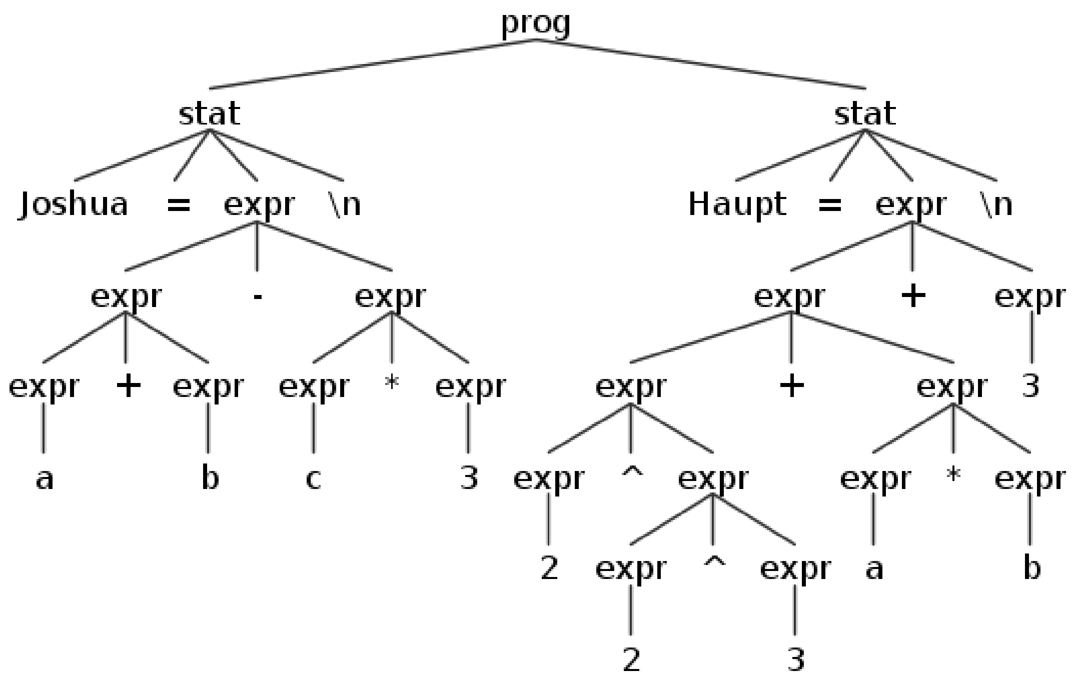
    Take screen shots of both DOS command window and GUI window (the one with the trees) and paste them to the next page.

A. Screenshot of command window containing ExprJoyRide and grun tests. (25 points)



```
(prog (stat Joshua = (expr (expr (expr a) + (expr b)) - (expr (expr c) * (expr 3))) \n) (stat Haupt = (expr 2) <missing NEW
LINE>) (stat (expr (expr (expr 2 3) + (expr (expr a) * (expr b))) + (expr 3)) \n))
josh@dragon-eos:~/Dev/CS330/HW4$ javac  *.java
josh@dragon-eos:~/Dev/CS330/HW4$ sudo cp antlr-4.5.1-complete.jar /usr/local/lib/
[sudo] password for josh:
josh@dragon-eos:~/Dev/CS330/HW4$ export CLASSPATH=".:/usr/local/lib/antlr-4.5.1-complete.jar:$CLASSPATH"
josh@dragon-eos:~/Dev/CS330/HW4$ alias antlr4='java -jar /usr/local/lib/antlr-4.5.1-complete.jar'
josh@dragon-eos:~/Dev/CS330/HW4$ alias grun='java org.antlr.v4.gui.TestRig'
josh@dragon-eos:~/Dev/CS330/HW4$ javac *.java
josh@dragon-eos:~/Dev/CS330/HW4$ java ExprJoyRide
Joshua=a+b-c*3
Haupt=2^2^3+a*b+3line 2:7 token recognition error at: '^'
line 2:9 token recognition error at: '^'
line 2:8 mismatched input '2' expecting {'*', '/', '+', '-', NEWLINE}
line 2:10 extraneous input '3' expecting {'*', '/', '+', '-', NEWLINE}
line 2:17 missing NEWLINE at '<EOF>'
(prog (stat Joshua = (expr (expr (expr a) + (expr b)) - (expr (expr c) * (expr 3))) \n) (stat Haupt = (expr 2) <missing NEW
LINE>) (stat (expr (expr (expr 2 3) + (expr (expr a) * (expr b))) + (expr 3)) <missing NEWLINE>))
josh@dragon-eos:~/Dev/CS330/HW4$ antlr4  Expr.g4
josh@dragon-eos:~/Dev/CS330/HW4$ javac *.java
josh@dragon-eos:~/Dev/CS330/HW4$ java ExprJoyRide
Joshua=a+b-c*3
Haupt=2^2^3+a*b+3
(prog (stat Joshua = (expr (expr (expr a) + (expr b)) - (expr (expr c) * (expr 3))) \n) (stat Haupt = (expr (expr (expr (ex
pr 2) ^ (expr (expr 2) ^ (expr 3))) + (expr (expr a) * (expr b))) + (expr 3)) \n))
josh@dragon-eos:~/Dev/CS330/HW4$ grun Expr prog -gui
Joshua=a+b-c*3
Haupt=2^2^3+a*b+3
```

B. Screenshot of gui window with the resulting tree. (30 points)



Answer the following questions as related to ANTLR Expr.g4 lab.

The expression rules in Expr.g4 is shown below:

```
expr:   expr ('*'|'/') expr
      |  expr ('+'|'-') expr
      |  INT
      |  ID
      |  '(' expr ')'
      ;
```

Even though this expression grammar is left-recursive (a big no no for most LL parsers) and ambiguous as we discussed in HW3, ANTLR4 is still able to process this grammar to create an unambiguous parser with correct precedence of * over +.

(a) Describe how ANTLR4 determine the operator precedence regarding ‘*’ and ‘/’ vs. ‘+’ and ‘–‘. (15 points)

If they are listed from top to bottom in the Expressions file, Antlr determines precedence by the the order they are listed in from top to bottom. Operators listed above will have higher precedence than those that are listed lower. If they are listed as a group on the same line such as expr ('*'|'/') expr then they have the same precedence.

(b) Describe how ANTLR4 determine the associativity of operators. (15 points)

By Default, ANTLR determines associates operates to the left. In right associativity is desired, it has to be declared in the following format: <assoc=right> expr ‘^’ expr the location of <assoc=right> varies depending on the version of ANTLR.

(c) Describe how ANTLR4 handles Left-recursive rule. (15 points)

ANTLR4 replaces left recursion with a (…)* that compares the precedence of the previous and next operators.

ANTLR4 can also only handle direct left recursion as indirect left recursion is not supported.

This assignment is worth 100 points and is due class time on Wednesday (10/19). Submit on moodle a zip file which contains your Expr.g4 and completed HW4ANTLR.doc. Late submission will be accepted until 12:00 PM Friday (10/21). Standard late penalty applies.