Lab 4: Flex/Lex, Bison/Yacc, and PLY (Python Lex and Yacc)
Due: April 9, 2023 (Sunday) 11:59pm CDT

The purpose of this lab is to practice using tools for lexical and syntactic analysis, and to gain some experience with Flex, Bison, and PLY (Python Lex and Yacc).

Useful info about Precedence in Yacc:
https://docs.oracle.com/cd/E19504-01/802-5880/6i9k05dh3/index.html

Part A: Flex and Bison (Lex and Yacc) – 60 points

Description:
You will start with a simple integer calculator. Its Flex and Bison specifications will be given to you, in the files called, respectively, icalc.l and icalc.y.

The two files have the specifications for operators '+' and '-'. Your task is to add '*', '/' (multiplication and division) and the unary minus operator to the specifications in icalc.l and icalc.y.

The - (unary minus) operator negates the value of the operand. For example, if quality has the value 100, -quality has the value -100.

Suggestion:
You may want to make the changes for '*', '/' (multiplication, division) operators first to make sure it works. Then you proceed to make the changes for the unary minus operator (-).

Hint for the unary minus operator:
You have to give a name to this unary minus operator – for example: %left NEG, near the location below in the icalc.y file

```
%left MINUS PLUS
```

Then, you need to add one more statement in the 'expr' location with a %prec keyword. You can research online for this. (prec – precedence)

```
expr:       expr PLUS expr        { $$ = $1 + $3; }
          | expr MINUS expr       { $$ = $1 - $3; }
          | LPAREN expr RPAREN              { $$ = $2; }
          | NUM               { $$ = $1; }
          ;
```

Installing Flex and Bison:
For Windows:
1. Download Flex 2.5.4a
2. Download Bison 2.4.1
    a. Goto (https://sourceforge.net/projects/gnuwin32/
    b. Select files:
    c. Find 'flex' and 'bison' there to download the software.
    d. Install the software.
    e. Copy the software to c:/GnuWin32/bin because Bison cannot handle the space in between the directory name.
    f. Add the path to the Environment Variables in Windows.

For Mac, you can use:

brew install flex
brew install bison
https://macappstore.org/flex/
https://macappstore.org/bison/
https://www.youtube.com/watch?v=SELYgZvAZbU
https://www.youtube.com/watch?v=cM_JGiXcMv0

Compilation & Execution of your Program:
1. Open Command prompt and switch to your working directory where you have stored your lex file (".l") and yacc file (".y")

2. Compiling Lex & Yacc file:
    i. flex icalc.l
       - lex.yy.c will be created
    ii. bison -dy icalc.y
       - y.tab.c and y.tab.h will be created
    iii. gcc lex.yy.c y.tab.c
       If you get the error below, rename y.tab.h to icalc.tab.h and recompile.
       (icalc.l:14:23: icalc.tab.h: No such file or directory)
       If it compiles successfully, a.exe will be created. (ignore the warnings.)
    iv. run the program – a.exe.
    v. type: 7+6, then it will show 13.

Use these test data:

Test1: 5-8*6-2
Test2: 3+5*8/2
Test3: -7 - 20/5
Test4: 70+2*(-30)

Copy and paste your output to a word doc. Zip your source programs and the executables. Do not zip your word doc. You can put the outputs of Part A and Part B in one Word Doc. Submit the word doc and the zipped file separately to canvas.

Part B: PLY (Python Lex and Yacc) – 40 points

PLY is a parsing tool written purely in Python. It is, in essence, a re-implementation of Lex and Yacc originally in C-language. It was written by David M. Beazley. PLY uses the same LALR parsing technique as Lex and Yacc.

Structure of a PLY file
PLY has the following two Python modules which are part of the ply package.
ply.lex - A re-implementation of Lex for lexical analysis
ply.yacc - A re-implementation of Yacc for parser creation
https://en.wikipedia.org/wiki/PLY_(Python_Lex-Yacc)

To install PLY on your computer, follow the steps outlined below:

1. Download the PLY source code from https://www.dabeaz.com/ply/ or from the lab4 folder on Canvas.
2. Unzip the downloaded zip file.
3. Navigate into the unzipped ply-3.10 folder
4. Run the following command in your terminal: python setup.py install

If you completed all the above, you should now be able to use the PLY module.

Description:
You are given a sample Python program file called 'ply_example1.py'. The Python program will import the lex and yacc modules and has the specifications for math expressions.

```
from ply import lex
import ply.yacc as yacc
```

It will parse a math expression and calculate the result.

Your tasks:
1. Fill in the missing code for the 'plus' operation in the sample program - it does not have the 'plus' operation code.
2. Correct the order of precedence. The sample program has the incorrect order of precedence. It will print '-12' as the result for '5-8*6-2'. Your task is to correct that.
3. Add code to get the math expression from the console and print the result. e.g.

```
Please enter a math expression or quit to exit:
5-8*6-2
The result is:  -45
:
5-8*6-2
The result is:  37
```

Use these test data (same as Part A):

Test1: 5-8*6-2
Test2: 3+5*8/2
Test3: -7 - 20/5
Test4: 70+2*(-30)

Copy and paste your output to a word doc and zip your source program. Do not zip your word doc. You can put the outputs of Part A and Part B in one Word Doc. Submit the word doc and the zipped file separately to canvas.