Programming Languages: Project #2 - Scanner and Parser

SWE3006 Programming Languages - Fall 2019

Due date: October 18 (Fri) 11:59pm

1 Prefix Expression Calculator

The goal of this project is to write a prefix calculator program using `lex` and `yacc`. The following grammar is for an infix calculator. In this project, you need to change this grammar for prefix expression.

```
List    -> List Stmt T_semicolon | Stmt T_semicolon
Stmt    -> Expr | Assgn
Assn    -> T_id '=' Stmt
Expr    -> Term | Expr T_add Term
Term    -> Factor | Term T_mult Factor
Factor  -> T_id | T_num | '-' Factor | '(' Expr ')'
T_add   -> '+' | '-'
T_mult  -> '*' | '/'
T_semicolon -> ';'
```

In a prefix calculator, you write an expression in different order. That is, "3 + 4" is written as "+ 3 4 " and "$a = 1$" is written as "$= a\ 1$ ".

Your parser must correctly handle associativity and precedence of operators according to the common convention.

2 Details

- Variable names consist of only alphabets, i.e., [a..zA..Z]
- Support negative integer numbers and real numbers
- Negative sign symbol is !, i.e., -100 is written as !100 in this calculator.
- Positive sign symbol is not supported.
- Calculation results should be in double, i.e., / 5 2 is 2.500000 not 2
- Use %lf format in printf
- Support expressions with numbers and IDs with parenthesis
- Print out resulting values for every expression
- Assignment statement does not print anything.
- If an uninitialized variable "abc" is used, print `Unknown variable: abc`
- For any other errors, your program may ignore the statement and continue, or you can stop processing the rest of statements.

3 Inputs and Outputs

Your program will read input from a file. I.e., the first argument of your command is the input file. The output must be printed to `stdout` as follows.

```
$ cat input.txt
   = a ! 100;
   + a * ! 2.1 34;
   * 10 (/ + 5.0 2 );
$ ./a.out input.txt
   -171.400000
   25.000000
$
```

More example inputs will be announced later to clarify requirements if necessary.

4 How to Submit

You will need to submit multiple files that you write for this project. Therefore, you should create a sub-directory, for example proj2, in your home, and then run pl_submit command as follows.

```
$ cd /home/2012345678/proj2
$ pl_submit proj2 ./
```

Note that you can submit multiple times. But only the last submission will be graded. Using the following command, you can check whether your file has been correctly submitted.

```
$ pl_check_submission proj2
```

Note that these commands only work in swin, swui, swye, swji machines. If you implemented your codes in your desktop, you must upload the file to these machines before you submit. Otherwise, pl_submit command will fail to read your implementation.

For any questions, please post them in Piazza so that we can share your questions and answers with other students and TAs. Please feel free to raise any issues and post any questions. Also, if you can answer other students' questions, you are welcome to do so. You will get some credits for posting questions and answering other students' questions.