CS 352 Spring 2015

Compiler Project Part 1

Posted Feb. 2, 2015, Due Feb. 12, 2015, 11:59pm

This project is to be done by each student individually. Read the Academic Honesty Policy posted on Piazza carefully.

1. Objective and Scope

This part of our compiler project is to implement a parser for a scripting language, called *miniscript*, which is similar to a subset of JavaScript. The following shows an example of miniscript.

```
<script type="text/JavaScript">
var two = 2 ; var ten = 10
var linebreak = "<br />"

document.write("two plus ten = ")
var result = two + ten
document.write(result)
document.write(linebreak)

result = ten * ten
document.write("ten * ten = ", result)
document.write(linebreak)

document.write("ten / two = ")
result = ten / two
document.write(result)
var ID
ID = result
document.write(linebreak)
document.write(ID)

</script>
```

The script may contain arbitrary white spaces, such as spaces, tabs, and new lines. These are to be skipped by the lexical analyzer unless they are in a quoted string. We assume that a quoted string never contains any other quotation marks. Line breaks are not allowed in the middle of a quoted string.

The first line and the last line will be in the exact form as in the example above. Between these two lines, there are zero or more statements. If a line contains a single statement, then it may or may not end with a semicolon. Otherwise, different statements are separated by semicolons.

We consider only three kinds of statements as shown in the example above:

1) Document write statements, in the exact form of *document.write(<param_list>)* where <param_list> represents zero or more parameters. Different parameters are separated by commas. Each parameter is either a quoted string, which never contains a linebreak (not to be confused by the identifier "linebreak" in the example), or an arithmetic expression. We only consider arithmetic operators +, -, *, /. Parentheses are allowed in an expression.

2) Assignment statements, with the left-hand side to be an identifier and the right-hand side to be either a quoted string or an arithmetic expression described in 1)

3) Declaration statements in the form of *var ID* or in the form of *var ID = <expr>* where <expr> is either a quoted string or an arithmetic expression described above

2. Requirement

You are to write a lex/flex program and a yacc/bison program to parse a *miniscript* program defined above, with no parsing conflicts reported by the yacc/bison tool. Do not use any precedence defining instructions such as %left to resolve parsing conflicts.

Do not add any output statements to the semantic actions in the yacc/bison program.

Submission instruction

1) No offline submission (such as email) is accepted.
2) Use the following command at CS lab machines, e.g. the XINU machines (i.e., xinu01.cs ~ xinu20.cs) to submit your homework.
turnin –c cs352 –p p1 [your working directory]

(Your home directories are shared amongst all CS lab machines.)

3) You MUST provide 'Makefile' for every programming question.
a. For example, your 'Makefile' of this assignment may look like

```
parser:y.tab.c lex.yy.c
  gcc y.tab.c lex.yy.c -o parser -lfl
y.tab.c : parser.y
  bison -y -d -g --verbose parser.y
lex.yy.c:parser.l
  lex parser.l
```

```
clean:
    rm -f lex.yy.c y.tab.c
```

NOTE that in the Makefile, a “tab” is required as the first character on each command line, e.g. “<tab> lex paser.l”. Otherwise you will see an error saying “missing separator”.

4) Your program MUST compile and run without any error at CS lab’s Linux machines. Please make sure your Makefile and program runs properly on such machines.
5) Make the final executable program’s name ‘parser’, as shown in the sample Makefile listed above. TA will run your program by executing:
> ./parser program_name
Where “program_name” is the file name containing the input program.

NOTE: Deviation from the above requirement will get a 10 point of penalty. (For example, if your code receives 90 points, the final score on the Blackboard for this assignment will be 80 points.)

You may find information on the following links posted on piazza to be useful, including the following:
http://ds9a.nl/lex-yacc/cvs/lex-yacc-howto.html