Homework 2

The written part (PART I) is due **September 25**, Thursday

The programming part (PART II) is due **September 27**, Saturday

On campus students must turn in a hard copy for the written part in class, with a cover page that has the student's name on it, which allows the grader to write the grade on the next page. **Distant –learning students must submit the written part on the Blackboard by midnight September 25th. (This is to ensure the submissions to be received by the teaching staff.)**

For the submission requirement for the programming part, see PART 2.

PART 1 (The written part)

(50 pts). In class we discussed function call statements. We assume that a function call to be in the form of

ID (<parameter_list>) ;

where <parameter_list> may be empty or may contain one or more IDs separated by commas, e.g.

abc10 ( ) ;

xyz (a) ;

nbc (I, j, k) ;

(1.1) Please write an SLR grammar for the function call statements described above.

(1.2) Build the LR(0) states and show the contents, i.e. the LR(0) items, in each state as we have done in the class. (Please review lecture notes.)

(1.3) Compute the Follow set for each nonterminal in your grammar

(1.4) Draw the SLR parsing table based on the results from (1.2) and (1.3)

(1.5) Apply the SLR parsing table on input "abc10 (a);" Show how the parsing stack and the next token position change with each parsing action, as shown in our lecture notes.

PART 2 (50 pts). We consider a very small form of Java script. The following shows an example of such a script.

```
<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

PART 2 is to write a lex/flex program and a yacc/bison program to parse a Javascript defined above, with no parsing conflicts reported. Do not use any precedence defining instructions as as %left to resolve parsing conflicts.

Submission instruction for PART 2

1) For programming assignment, no offline submission will be 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 cs502 –p pa2 [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

```
hw2:y.tab.c lex.yy.c
    gcc y.tab.c lex.yy.c -o hw2 -lfl
y.tab.c : hw2.y
    bison -y -d -g --verbose hw2.y
lex.yy.c:hw2.l
    lex hw2.l
clean:
    rm -f lex.yy.c y.tab.c
```

4) Your program MUST compile and run without any error at XINU machines. Please make sure your Makefile and program runs properly at XINU machines.
5) Any special instruction (if needed) for running your program should be included in the file named 'README'. By default, TA assumes your program could be run like this:
> ./hw2 program_name
Where hw2 is your excusable file after compilation, and "program_name" is the file name containing the input program.

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