CS352 Homework 3 (Due 11:59pm, Feb. 20th, on GradeScope)

IMPORTANT NOTES:

(1) Unless announced otherwise, submissions of written homework assignments must follow the guidelines given in Homework 1. For each **state diagram** you are asked to draw, please follow **the required formats** as stated in the lectures.
(2) Every grammar given in the problems already has a start nonterminal that does not appear in the right-hand side. **Do not introduce a new start nonterminal.**
(3) In each parsing table to be constructed by the student, the **columns** must be listed in the **order** specified in the problem, so as to make grading easier. There is a penalty for not following the order.

Problem 1 (25 Pts)

You are given the following grammar for C-like structures:

1. <decl> → struct <tag> “{” L “}”
2. <tag> → ID
3. <tag> → ε
4. L → L <field>
5. L → <field>
6. <field> → int ID
7. <field> → ID ID

This grammar has five kinds of terminals: ID, int, “{“, “}”, and struct. Please draw the LR(0) state diagram and try to build the SLR parsing table for the given grammar. If you find any parsing conflicts (shift/reduce, or reduce/reduce), please circle them in the parsing table. (For this problem, reduce actions should be marked **only** in the columns that are marked by terminals belonging to the **FOLLOW(A)**, where A is the nonterminal being reduced to.)

The required column ordering is shown below.

	struct	{	}	ID	int	$	<tag>	L	<field>

Problem 2 (25 pts).

We have the grammar on next page, with rules 1-10 and semantic actions written in YACC format. YACC adds rule 0 shown below. (The terminals are: $end, ‘\n’, ‘(‘, ‘)’, ‘*’, ‘+’, ‘-‘, ‘/’, INT).

The parsing table shown after the grammar is formatted based on the state diagram given by YACC. Each blue horizontal bar marks all columns in which the marked action applies. For example, in row 0, r2 is applied to all columns from Int to ‘\n’. Due to the limited space, the letter ‘s’ for ‘shift’ is omitted for shifting actions, and only the target state is listed. For example, ‘3’ in row 1 under ‘Int’ means “shift 3”.

Assume input 100 – 30 * (50 + 3) \n $end, please follow the format in the lecture slides to **show the parsing stack after each shift, reduce, and goto action. Whenever the semantic stack changes its content, please also show its content.** If the input is incorrect, show the stacks as far as possible until the parser detects an error. *See the format of the solution on next page.*

```
(0)  $accept: program $end   ;

(1) program: program expr '\n'
                { printf("%d\n", $2); }
 (2)      |   ;
expr:
(3)   expr '+' term   {$$ = $1 + $3; }
(4)   | expr '-' term {$$ = $1 - $3; }
(5)   | term   ;
```

```
term:
(6)    factor
(7)   | term '*' factor {$$ = $1 * $3; }
(8)   | term '/' factor {$$ = $1 / $3; }
         ;
factor:
(9)   '(' expr ')'    {$$ = $2;}
(10)    |  Int          {$$ = $1;}
```

	Int	(	)	$end	<eof>	+	-	*	/	\n	Expr	Term	Factor	Prog
0	r2													1
1	3	4		2							5	6	7	
2					Accept									
3	r10													
4	3	4									8	6	7	
5						10	11			9				
6	r5							12	13					
7	r6													
8			14			10	11							
9	r1													
10	3	4										15	7	
11	3	4										16	7	
12	3	4											17	
13	3	4											18	
14	r9													
15	r3							12	13					
16	r4							12	13					
17	r7													
18	r8													

In the parsing table listed above, <eof> is the same as $ in the lectures, not to be confused with token $end.

The solution should be entered in the following tabular format:

Parsing Stack (top is the rightmost)	Next Action	Remaining Input
0	in the form of s?, r?, or g?	100 – 30 * (50 + 3) \n $end
? the target state after this action		(show remaining input again only if it is changed)
Content of semantic stack if changed	Semantic Stack	

Enter the words "Semantic Stack" to indicate the first column is the content of the semantic stack

Problem 3 (25 pts)

You are given the following grammar:

1. <prog> →<limit> L <limit>
2. <limit> → lines
3. <limit> →ε
4. L → L lines stmt
5. L →L ; stmt
6. L → stmt

Note that this grammar has three kinds of terminals: lines, ';' and stmt. **Construct an LR(0) state diagram and try to build the SLR parsing table. Circle parsing conflicts if you find any.** The column ordering for the parsing table built in Problem 3 must be as follows:

	stmt	lines	;	$	<limit>	L

Problem 4 (25 pts)

For the grammar in Problem 3, **Construct an LALR(1) state diagram and try to build the LALR(1) parsing table. Circle parsing conflicts if you find any.**

Reminder: With LALR(1), LR(1) states that have the same set of LR(0) items but differ in lookahead must be merged. This may introduce new transitions from some states already constructed. If you run YACC to obtain a state diagram for comparison, remember that YACC may perform reduce under tokens that do not belong to the lookahead set. Your solution must follow the algorithm in the lecture for LALR(1) and **perform reduce only for those tokens in the lookahead set**. The column ordering for the parsing table built in Problem 4 must be the same as in Problem 3.