Midterm Exam for Compiler
by Prof. Jenq-Kuen Lee

1. (40%) If we use BNF form to write a grammar for an arithmetic expression includes "+" (addition) and parenthesis. We get a grammar below:

```
E ->  E  +  E
E ->  ( E )
E ->  Number
```

Assume the precedence order from the highest to the lowest is parenthesis, "+". The addition operator is left associate.

(a) Re-Write the above grammar into an un-ambiguous grammar following the given precedences and associativity.

(b) Is the grammar generated in (a) a LL(1) grammar? If it's not a LL(1) grammar, try to convert it into a LL(1) grammar?

(c) To use the concept of selection set to explain why the grammars you generated in (b) is a LL(1) grammar.

(d) Write a LL(1) parser based on the grammar and results in (b) and (c). Note that a top-down recursive program should be used here.

(e) Write a LL(1) code generator for the grammar generated in (b) and (c). You can choose the pseudo codes in your definitions for target code generations. The execution of your generated codes should be in the same meanings as arithmetic calculations.

2. (16%) (a) To write a Lex-style regular expression to represent the syntax of the constant of a floating number in C language.

(b) To write a Lex-style regular expression to represent the syntax of the variable name in Java program.

3. (8%) Construct the minimum-state DFA for the following regular expression.

$(a|b)^*b(a|b)$

4. (8%) Can any regular expression be converted into an equivalent deterministic finite automata (DFA)? Briefly explain the procedures if it can be done.

5. (8%) Explain why LL(1) parsers or top-down parsers do no allow left- recursive grammars.

6. (8%) Explain what LL(1) grammar is?

7. (12%) Describe the languages denoted by the following regular expressions.

(a) $0(0|1)^*0$

(b) $(0|1)^*0(0|1)(0|1)$