Final Exam for Compiler
by Prof. Jenq-Kuen Lee

1. (20%)

   Construct an SLR(1) parsing table for the grammar below.

```
E -> E sub R  |  E sup E   |    { E }  |  c
R -> E sup E  |  E
```

2. (20%)

   (a) In the final project of our class, we use YACC to generate bytecode assembly codes for a Java programs. Explain the code generation techniques when handling a Java expression. We give a simple grammar of Java expression below. The operator precedence and associativity follow Java languages.

```
E -> E + E
E -> E * E
E -> E = E
E -> (E)
E -> NUMBER
E -> Identifier
```

   (b) The grammar above is an ambiguous grammar. Explain the techniques used by YACC to handle grammar ambiguities generated by different operators.

3. (20%)

   Write a YACC program that will take integer arithmetic expressions as input and produce a bytecode output which the execution of the bytecodes will be the value of the input arithmetic expression. You can also use x86 instead of bytecodes in your solution.

4. (10%)

   Write a YACC-like grammar for function and function body.

5. (30%) Explain the relationships among the set of the bottom-up parsing schemes given below. Please use follow_set or look_ahead set concepts to illustrate the idea.

   (a) Give an example grammar which is in LR(1) but not in LALR(1). r-r

   (b) Give an example grammar which is in LALR(1) but not in SLR(1). s-r

   (c) Give an example grammar which is in SLR(1) but not in LR(0). X