Compiler Design Final 0A June 16, 2000

1. [20] Consider the following grammar G:

$$S \rightarrow (L) \mid a$$
$$L \rightarrow L, S \mid S$$

S → (L) | a
L → L, a | a
L S
L · S
L → aL' |
L' → , a L'
L · S.

(a) Eliminate the left-recursion of G.
(b) Construct a predictive parser for G. Table
(c) Show the behavior of the parser on the sentence (a, a).

2. [20] Consider the following grammar G:

L → L, (L) | (L)

$$S \rightarrow AaAb \mid BbBa$$
$$A \rightarrow \epsilon$$
$$B \rightarrow \epsilon$$

Follow (αA)
First (β a)
a

(a) Is G LL(1)? If yes, give the parse table. Otherwise, show why.
(b) Is G SLR(1)? If yes, give the parse table. Otherwise, show why.

3. [30] Consider the following grammar G:

$$S' \rightarrow S$$
$$S \rightarrow Xa \mid Yb \mid bXb$$
$$X \rightarrow c$$
$$Y \rightarrow c$$

Follow(X): a, b
Y: (b)
First(X): c
First(Y): c

(a) Is G SLR(1)? If yes, give the parse table. Otherwise, show why.
(b) Is G LR(1)? If yes, give the parse table. Otherwise, show why.
(c) Is G LALR(1)? If yes, give the parse table. Otherwise, show why.

4. [10] Translate the expression $-(a + b) * (c + d) + (a + b + c)$ into

(a) a syntax tree
(b) postfix notation
(c) three-address code

a + b + c +

5. [10] In C, the for statement has the following form

c - b + d =

for ($expr_1$; $expr_2$; $expr_3$) stmt

It has the same meaning as

a + b + c / d + e

$expr_1$;
while ($expr_2$) **do begin** stmt; $expr_3$ **end**

Write down the actions to translate C-style **for** statement into three-address code.

6. [10] Write down a grammar and its actions to translate infix expressions into postfix form.
Note that the operators +, −, *, and / are left-associative and the operators * and / have higher precedence.