CS 164, Fall 2000
Midterm #1

Problem #1

First and Follow Sets (15 points)
Consider the following three grammars

1.
A -> BC
B -> Ax | x
C -> yC | y

2.
A -> BC
B -> Ax | x | epsilon
C -> yC | y

3.
A -> BC
B -> Ax | x | epsilon
C -> yC | y | epsilon

For each of the following statements about First and Follow sets, indicate for which grammars(s) (1, 2, or 3) the statement is correct. Each statement is correct for one or more grammars-be sure to list all the grammars for which a statement is correct.

Note that grammar 3 is a small extension of grammar 2, which is a small extension of grammar 1. You may find it helpful to compute the First and Follow sets beginning with the simplest grammar.

- First(A) = {x, y}

- Follow(A) = {$,x}

- Follow(B) = {$,x,y}

- First(C) = {y}

- Follow(C) = {$,x}

Problem #2

Regular Expressions and Context-Free Grammar (15 points)

In this problem we will show that every regular language is also a context-free language. The proof is to construct for every regular expression an equivalent context-free grammar. A regular expression is equivalent to a context-free grammar if they generate the same language.

We will do the first case for you; the problem is to show how to do two other cases. Consider the regular expression AB. Assume that we have already converted A to an equivalent context-free grammar with start symbol SA and that we have converted B to an equivalent context-free grammar with start symbol SB. Then a context free grammar for the language AB with start symbol SAB is

(We assume that the grammars for Sa and SB share no non-terminals.)

(a) In the same style as above, what is the context-free grammar for A + B?

(b) What is the context-free grammar for A*?

Problem #3

LR Parsing (25 points)

In these problems we describe a DFA for accepting viable prefixes of a grammar. Your assignment is to give a grammar whose DFA recognizing viable prefixes satisfies the description.

Assume the DFA's we describe contain LR(0) items-no lookaheads are involved. Do not augment your grammar with an extra production S -> S'. Just find any grammar that works.

(a) The DFA consists of one state and no transitions.

(b) The DFA consists of two states s1 and s2 and there is one transition from the start state s1 to s2.

(c) The DFA consists for three states s1, s2, and s3. There is one transition from the start state s1 to s2, one transition from s2 to itself, and one transition from s2 to s3.

Problem #4

Syntax-Directed Translation (10 points)

For the following grammar for arithmetic expression define a syntax-directed translation that records the sign (POS or NEG) of each subexpression E in the attribute E sign; Write your semantic actions to the right of the corresponding production. you may assume that all integers are non-zero (so you don't need to worry about

the sign of zero).

E -> unsigned_integer

| +E1

| -E1

| E1*E2

Problem #5

LL Parsing (25 points)

Consider the following grammar:

S -> aS | Ab

A -> XYZ | epsilon

X -> cS | epsilon

Y -> dS | epsilon

Z -> eS

(a) Give the LL(1) parsing table for this grammar.

(b) Give the leftmost derivation of the string aebb.

(c) Show that if we add the production X -> bS that the grammar is no longer LL(1).

Problem #6

Error Recovery (10 points)

Consider the following grammar for strings of ab's

A -> abA | epsilon

(a) Augment the grammar with productions that allow a parser to recognize all erroneous strings as well as valid strings. The non-terminal ERROR should derive the suffix starting at the first erroneous terminal. More precisely, in the case that the input is erroneous, then ERROR should

- derive a suffix of a of the string such that if s is deleted, then the string is in the original language, and

- derive the shortest such suffix.

You may use as many new productions and non-terminals besides ERROR as you like. Assume that a and b are the only possible terminals.

(b) What language does your augmented grammar generate?
