CS 132, Jens Palsberg Nov 2, 2005

Midterm Exam

Each of the three questions is worth five points, for a total of 15 points. Write your name and id number at the top of the first page you submit. Write the solutions for different questions on different sheets of paper. Don't submit this handout.

1. Consider the grammar

$$\begin{array}{lcl} A & ::= & B\ x \mid B\ y \\ B & ::= & A\ y \mid x \end{array}$$

where $\{A, B\}$ is the set of nonterminal symbols, A is the start symbol, and $\{x, y\}$ is the set of terminal symbols. Give an LL(1) grammar which generates the same language as the one above, show the FIRST and FOLLOW sets for each nonterminal symbol in the LL(1) grammar, and the show predictive parsing table for the LL(1) grammar. Argue that the grammar is LL(1).

2. Consider the grammar

$$\begin{array}{lcl} A & ::= & x\ B \mid C\ z \\ B & ::= & A\ x\ C \\ C & ::= & y\ B \mid \epsilon \end{array}$$

where $\{A, B, C\}$ is the set of nonterminal symbols, A is the start symbol, $\{x, y, z\}$ is the set of terminal symbols, and ϵ denotes the empty string. Is the grammar LL(1)? Justify your answer. As part of your answer, show the FIRST and FOLLOW sets for each nonterminal symbol.

3. Consider the grammar

$$\begin{array}{lcl} A & ::= & x\ C \mid z \\ B & ::= & y\ A \\ C & ::= & B\ x \mid A\ y\ B \end{array}$$

where $\{A, B, C\}$ is the set of nonterminal symbols, A is the start symbol, and $\{x, y, z\}$ is the set of terminal symbols. The grammar is LL(1). Sketch Java code in reasonable detail for a recursive-descent parser based on the grammar.

CS 132 Fall 2005, Midterm Solution

Question 1

```
A ::= x C D
C ::= x | y
D ::= y C D | epsilon
```

	First	Follow	Nullable	parsing table x	y	$
A	{x}	{$}	No	xCD	-	-
C	{x,y}	{$,y}	No	x	y	-
D	{y}	{$}	Yes	-	yCD	-

Question 2

	First	Follow	Nullable
A	{x,y,z}	{$,x}	No
B	{x,y,z}	{$,x,z}	No
C	{y}	{$,x,z}	Yes

The grammar is LL(1) because (i) the two options for A have disjoint First sets and (ii) for C, the First set for (y B) is disjoint from Follow(C).

Question 3

Here is a sketch of the Java code:

```
   void A() {
      if ( token == 'x' ) {
          get-next-token();
          C();
      }
      else {
          if ( token == 'z' ) { get-next-token(); }
          else { error(); }
      }
   }

   void B() {
      if ( token == 'y' ) { get-next-token(); }
      else { error(); }
      A();
   }

   void C() {
      if ( token == 'y' ) {
          B();
          if ( token == 'x' ) { get-next-token(); }
          else { error(); }
      }
      else {
          A();
          if ( token == 'y' ) { get-next-token(); }
          else { error(); }
          B();
      }
   }
```