

Name: _____

USC loginid (no SSN): _____

CS 410 Midterm Exam
Fall 2003 [Bono]
October 20, 2003

There are 6 problems on the exam, with 50 points total available. There are 8 pages to the exam, including this one; make sure you have all of them. If you need additional space to write any answers, you may use the backs of pages (just direct us to look there).

Put your name and loginid at the top of the exam. Please read over the whole test before beginning. Good luck!

	value	score
Problem 1	5 pts.	
Problem 2	10 pts.	
Problem 3A	9 pts.	
Problem 3B	6 pts.	
Problem 4	5 pts.	
Problem 5	10 pts.	
Problem 6	5 pts.	
TOTAL	50 pts.	

Problem 1 [5 pts.]

Write a regular expression equivalent to the following grammar for assignment statements. I.e., the regular expression you write will denote the same language as the grammar below does. The terminals are { **id**, **#**, **+**, **=** } (You can think of the **#** as a dereferencing operator. We didn't use ***** to avoid confusion.)

S → L = R

L → # L
| id

R → id
| R + id

Problem 2 [10 pts.]

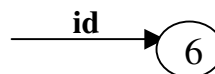
Use subset construction to build a DFA equivalent to the NFA below. Show your work. Note: ϵ is the epsilon symbol.

	a	b	ϵ
1	{ 2, 5 }	{ 1, 3 }	{ }
2	{ 2 }	{ 3 }	{ 4 }
3	{ 5 }	{ }	{ }
4	{ }	{ 4 }	{ 3 }
<u>5</u>	{ }	{ }	{ }

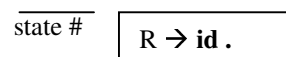
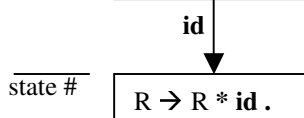
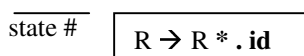
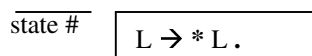
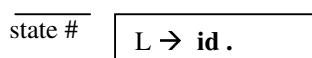
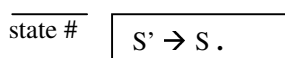
Problem 3 [15 pts. total]

Part A [9]. Build the DFA for recognizing viable prefixes for the grammar below. Show the set of items for each state (except ones we provide, see below). Note: the grammar has already been augmented for you. You may use the space below, or the back of the previous page (just direct us to look there).

To save you some work, we have created some of the states for you below. For those ones, just number them, and in your state diagram, you can just refer to them by number instead trying to make transitions to our “blobs” below, so you don’t get a lot of crossed lines. E.g.,



- 0) $S' \rightarrow S$
- 1) $S \rightarrow L = R$
- 2) $L \rightarrow * L$
- 3) $L \rightarrow id$
- 4) $R \rightarrow R * id$
- 5) $R \rightarrow id$



Problem 4 [5 pts]

Show that the following grammar is ambiguous. The ϵ below is epsilon.

$$S \rightarrow A B A B$$

$$A \rightarrow \mathbf{a} A$$

$$| \epsilon$$

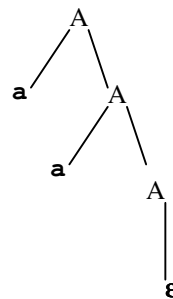
$$B \rightarrow \mathbf{b} B$$

$$| \epsilon$$

Reminder: $A \rightarrow \epsilon$ is an epsilon production. Here is a sample parse tree using a grammar that has an epsilon production:

grammar: $A \rightarrow \mathbf{a} A$
 $| \epsilon$

parse tree for **aa**:



Problem 5 [10 pts.]

Write the grammar and semantic rules to build an AST for a comma-separated list of one or more expressions. We provide the list-building primitives below.

Here are three sample sentential forms in the language:

expr *expr* , *expr* *expr* , *expr* , *expr*

where *expr* could expand to some expression.

Use the non-terminal *list* as the start symbol of your grammar. Use the attribute *list.tree*, which is type `ExprList`, for the AST.

The assumptions you can make:

- You may assume elsewhere in the grammar there exist rules for non-terminal *expr*, which compute the attribute *expr.tree*. *expr.tree* has type `Expr`.
- Use the following list-building primitives:
 - `ExprList makeEmpty(); // returns an empty expression list`
 - `ExprList makeList(ExprList smallList, Expr last);`
// Returns a list that has "last" appended to the end
// of "smallList".
// Thus the resulting list is one expr longer than "smallList"

Problem 6 [5 pts.]

A symbol table class with the following operations would be useful for implementing the most-closely-nested scoping rule. In the headers below, assume *id* is an identifier.

```
void enterscope()
```

Enter a new nested scope.

```
void exitscope()
```

Exit nested scope, returning to the previous (outer) scope.

```
Type * probe(id)
```

Look up *id* in the innermost scope only. Returns NULL if not found.

```
Type * lookup(id)
```

Look up *id*, searching scopes outwards. Returns NULL if not found.

```
void addid(id, type)
```

Add the pair $\langle id, type \rangle$ to the current scope.

If we're doing semantic analysis on a program whose variables follow the most-closely-nested scoping rule, for each of the places in a program listed below:

- give the symbol table operation(s) that would be invoked at that point,
- describe a semantic error that could be detected at that point based on the results of the operation(s)

[There is more space here than you will need to answer the question. Each description should only be about one sentence.]

Place 1: Declaration of a variable.

Place 2: Reference to a variable (i.e., variable use site).