

Name: \_\_\_\_\_

USC ID: \_\_\_\_\_

## Midterm Exam CS 410, Fall 1999 [Bono]

October 21, 1999

There are 7 problems on the exam, with 80 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 where to look).

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

	<b>value</b>	<b>score</b>
Problem 1	10 pts.	
Problem 2	12 pts.	
Problem 3	14 pts.	
Problem 4	10 pts.	
Problem 5	14 pts.	
Problem 6	10 pts.	
Problem 7	10 pts.	
<b>TOTAL</b>	80 pts.	

## Problem 1 [10 pts.]

**Part A.** An NFA is different than a DFA in that it allows

transitions on \_\_\_\_\_

and transitions to \_\_\_\_\_.

**Part B.** Thompson's construction is a method for automatically building a

\_\_\_\_\_

from a \_\_\_\_\_.

**Part C.** Give a regular expression equivalent to

$m^+$

which does not use the + operator:

**Part D.** In our compiler project we built the following type of parser (circle one):

- a) DFA
- b) LR
- c) semantic
- d) LL

**Part E.** The dangling-else problem causes (circle one):

- a) the lexical analyzer to be unable to recognize certain keywords
- b) difficulty left-factoring the grammar
- c) it to be impossible to create semantic rules for building if-else ASTs
- d) a shift-reduce conflict in an LR parser
- e) dandruff

## Problem 2 [12 pts.]

Find the FIRST and FOLLOW sets for each of the non-terminals in the following grammar. Show your work.

(in the grammar below  $\epsilon$  denotes epsilon, the empty string)

$$\begin{aligned} S &\rightarrow a B c \\ &\quad | a S Y c \\ &\quad | m \\ Y &\rightarrow d Y Y \\ &\quad | g Y \\ &\quad | B \\ B &\rightarrow x B \\ &\quad | \epsilon \end{aligned}$$

**Problem 3 [14 pts.]**

**Part A [10].** Use subset construction to build a DFA equivalent to the NFA below. Show your work.

	<b>a</b>	<b>b</b>	$\epsilon$
<b>1</b>	{ }	{ }	{ 2, 5 }
<b>2</b>	{ 3 }	{ }	{ }
<b>3</b>	{ }	{ 4 }	{ }
<b>4</b>	{ }	{ }	{ 2, 5 }
<b>5</b>	{ }	{ }	{ }

**Part B [4].** Give a regular expression for the language recognized by the DFA (or NFA) from part A

### Problem 4 [10 pts.]

Show that the following grammar is ambiguous.

$$\begin{aligned} Y &\rightarrow d Y Y \\ &\quad | g Y \\ &\quad | B \\ B &\rightarrow x B \\ &\quad | \varepsilon \end{aligned}$$

## Problem 5 [14 pts.]

For a code pretty-printer, there might be a maximum number of levels of nesting of blocks allowed for display purposes, to avoid wrapping around the end of the page or display as we indent some for each level of nesting. For the following simplified version of a C program grammar, write semantic rules to compute the attribute `P.tooDeep`, which is true iff the deepest nesting level of blocks in the program is larger than `MAXNEST` (an integer constant). Assume the function body itself is the outermost block (e.g., the first occurrence of `SIMPLE` in the example is at nesting depth 1). Hint: you may use additional semantic attributes.

Here's an example string generated by this grammar shown pretty-printed (deepest nesting is 3):

```
FUNC_HDR {
  SIMPLE ;
}
FUNC_HDR {
  SIMPLE ;
  SIMPLE ;
  {
    SIMPLE ;
    {
      SIMPLE ;
    }
    SIMPLE ;
  }
}
FUNC_HDR {
  SIMPLE ;
  {
    SIMPLE ;
  }
}
```

---

Grammar (with nonterms: `P FList SList S` and terms: `FUNC_HDR { } ; SIMPLE`):

`P` → `FList`

`FList` → `FUNC_HDR { SList }`  
| `FList1 FUNC_HDR { SList }`

`SList` → `S`  
| `S SList1`

`S` → `SIMPLE ;`  
| `{ SList }`

### Problem 6 [10 pts.]

Consider the following grammar *which has already been augmented*.

Show the set of LR(0) items corresponding to the start state for a SLR parser. Also show all the sets of items reachable from the start state in one `goto` step. Show the `goto` transitions from the start state and to the other states you gave. [Note: the resulting collection of item sets is a subset of the canonical collection – not all of the sets in the collection.]

- 0)  $L' \rightarrow L$
- 1)  $L \rightarrow L M$
- 2)  $L \rightarrow M$
- 3)  $M \rightarrow x$
- 4)  $M \rightarrow \{ L \}$

**Problem 7 [10 pts.]**

Build the LL(1) parse table for the following grammar. Show your work.

$$L \rightarrow M L$$
$$L \rightarrow M$$
$$M \rightarrow x$$
$$M \rightarrow \{ L \}$$