

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

USC loginid (e.g., ttrojan): \_\_\_\_\_

**CS 410 Midterm Exam**  
**Fall 2005 [Bono]**  
October 17, 2005

There are 8 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 to look there).

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

	<b>value</b>	<b>score</b>
Problem 1	18 pts.	
Problem 2 & 3	10 pts.	
Problem 4	10 pts.	
Problem 5	14 pts.	
Problem 6	5 pts.	
Problem 7	13 pts.	
Problem 8	10 pts.	
<b>TOTAL</b>	80 pts.	

## Problem 1 [18 pts.]

Short answer problems. Point values shown in brackets.

A. [2] *bison* is a tool for automatically building a \_\_\_\_\_ from a \_\_\_\_\_.

B. [2] A DFA is a machine to \_\_\_\_\_.

C. [2] A parser is a machine to \_\_\_\_\_.

D. [1] A regular expression denotes \_\_\_\_\_.

E. [1] A context free grammar denotes \_\_\_\_\_.

F. [2] Among other operations, symbol tables generally need an operation to check if a variable is defined in the innermost scope (i.e., one that would not check other outer scopes). Why?

(N = nondeterministic; D = deterministic; FA = finite automaton; PDA = pushdown automaton)

G. [1] Is an NFA a more powerful machine than a DFA? \_\_\_\_\_

H. [1] Is an PDA (a.k.a., NPDA) a more powerful machine than a DPDA? \_\_\_\_\_

I. [1] Are there things that can be recognized with a DFA that can't be recognized with a PDA? \_\_\_\_\_

J. [5] In a language such as C++, where identifiers have to be declared before they are used, how many passes are necessary to do semantic analysis and why?

## Problem 2 [5 pts.]

Using Thompson's Construction, build an NFA equivalent to the regular expression given below:

$a \mid a^*b$

**Problem 3 [5 pts.]**

Given the alphabet  $\{a, b\}$  write a regular expression for the set of all strings such that every  $a$  in the string is immediately followed by at least two  $b$ 's.

**Problem 4 [10 pts.]**

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

	<b>a</b>	<b>b</b>	<b><math>\epsilon</math></b>
<b>1</b>	{ 1 }	{ 3 }	{ }
<b><u>2</u></b>	{ }	{ }	{ 4 }
<b>3</b>	{ 2 }	{ 4 }	{ 2 }
<b>4</b>	{ 4 }	{ 2, 4 }	{ }

### Problem 5 [14 pts.]

**Part A** [10]. Give the FIRST and FOLLOW sets for each of the non-terminals in the following augmented grammar:

$$\begin{aligned} S' &\rightarrow S \\ S &\rightarrow S B x \\ &\quad | y \\ B &\rightarrow m B r \\ &\quad | r \end{aligned}$$

**Part B** [2]. Consider the DFA to recognize viable prefixes for an SLR parser for this grammar. Give an example *item* from the grammar above that will be part of a state that will have reduce entries. Note: Do not construct the whole DFA or even a whole set of items to answer this problem.

**Part C** [2]. Using your answer from part B, name which set you computed in part A would be used to figure out the reduce entries for the state with the item from part B.

### Problem 6 [5 pts.]

Show that the following grammar is ambiguous.

$$\begin{aligned} S &\rightarrow x S x \\ &\quad | x T x \end{aligned}$$

$$\begin{aligned} T &\rightarrow T x \\ &\quad | x \end{aligned}$$

### Problem 7 [13 pts.]

Build the DFA for recognizing viable prefixes for an SLR parser for the grammar below. Show the set of items for each state. Note: the grammar has already been augmented for you.

0)  $S' \rightarrow S$

1)  $S \rightarrow x S y$

2)  $S \rightarrow x T y$

3)  $T \rightarrow T z$

4)  $T \rightarrow z$

### Problem 8 [10 pts.]

Write semantic rules for the grammar below to compute the boolean semantic attribute  $S.moreb$ , which is true iff the sentence parsed has more b's in it than a's. Note: the subscripts below are only to distinguish multiple instances of a non-terminal in a production.

Hint: you can use additional semantic attributes.

You may not use any global variables in your answer.

$S \rightarrow A$

$A \rightarrow a B d$

$A \rightarrow a A_1 d$

$B \rightarrow b c$

$B \rightarrow b B_1 c$