CS 431 Translation of Computer Languages **Fall 1999**

Exam I

Given: 28 February 2002 Due: 12 March 2002, start of class

You may use any books or notes you wish for this exam. Solutions will be graded on a curve; partial credit will be given where sufficient detail is provided.

YOUR ANSWERS MUST BE PROVIDED ON THE PAGES OF THIS EXAM. Turn in other pages if you wish to show your work, but I WILL ONLY GRADE ANSWERS WRITTEN ON THIS EXAM.

Your exam is due at the start of class (4:07 PM) in Lopata 101.

By turning in this exam, you agree that you have read, understood, and abided by the Statement of Student Academic Integrity for the purposes of this exam.

Name:		
Student ID:		
Problem Number	Possible Points	Received Points
1	20	
2	20	
3	20	
4	20	
5	20	
Total	100	

1. (20 points) With regard to its DNA, a species can be considered as a language over the alphabet $\Sigma = \{ a, c, t, g \}$. A string s of DNA is said to be *complementary palindromic* if

$$Complement(Rev(s)) = s$$

where $Rev(s)$ is the reverse of string s and $Complement(s)$ is a character-substitution mapping, defined as follows:

$$\begin{aligned} Complement(a) &= t \\ Complement(t) &= a \\ Complement(c) &= g \\ Complement(g) &= c \end{aligned}$$

Thus, the string `gaattc` is complimentary palindromic:

- Its reverse is `cttaag`
- The reverse string's complement is `gaattc`

Other examples include `aaaatttt`, `actgcagt`.

The following is offered as a grammar for Complementary Palindromic DNA Strings:

$$\begin{array}{lcl} \text{S} & \rightarrow & \text{a S t} \\ & | & \text{t S a} \\ & | & \text{c S g} \\ & | & \text{g S c} \\ & | & \lambda \end{array}$$

(a) (4 points) Show a parse tree for the string "g a a t t c"

Continued on next page...

(b) (4 points) Show the predict sets for each rule for this grammar.

S → a S t	
S → t S a	
S → c S g	
S → g S c	
S → λ	

(c) (4 points) Explain why the grammar is or is not ambiguous.

Continued on next page...

(d) (4 points) Explain why the grammar is or is not $LL(1)$ parsable.

(e) (4 points) We now augment strings in the language above to include a marker `x` in the middle of the string (the numbers in parentheses are production numbers to help answer this question);

$$
\begin{array}{rcl}
S & \rightarrow & a\ S\ t\ (1) \\
 & | & t\ S\ a\ (2) \\
 & | & c\ S\ g\ (3) \\
 & | & g\ S\ c\ (4) \\
 & | & x\ \quad (5)
\end{array}
$$

Fill in the following $LL(1)$ parse table for this grammar. If predict sets are not disjoint, then write multiple symbols into the appropriate box.

NonTerm	a	c	t	g	x
S					

2. (20 points) Consider the following grammar:

P	→	Stmt $
Stmt	→	MethodDecl
	|	MethodCall
MethodDecl	→	MethodId Parameters semicolon
MethodCall	→	CallId Parameters x
MethodId	→	id
CallId	→	id
Parameters	→	lparen Params rparen
Params	→	Params id
	|	id

(a) Below, and on the next page if necessary, show your LR(0) construction for this grammar. For your convenience, this grammar is available online and can be installed in your CEC workspace by:

```
make -f ~cs431/ExamGrammar/Makefile Setup
```

You can see the LR(0) collection of states after installation by:

```
make states
```

You may use the online copy to check your solution, but you must turn in your original solution—do not just copy the machine you get from CUP's analysis.

Continued on next page...

(b) The grammar is not LR(0). Which state(s) has/have a conflict?

Continued on next page...

(c) Is the grammar ambiguous? If so, show two parse trees for the same string. Otherwise, explain why it is not ambiguous.

(d) Would more lookahead resolve the conflict? If so, state how much lookahead is needed. If not, explain why no amount of lookahead suffices.

3. (20 points) True or False? Where I ask you to "explain why" I expect you to do one of the following. If you believe the statement is false, then you must give a counterexample to show that. If you believe the statement is true, you must give a logical, convincing argument (a proof would be nice). You will not get much credit if you simply given an example to demonstrate the truth of the statement.

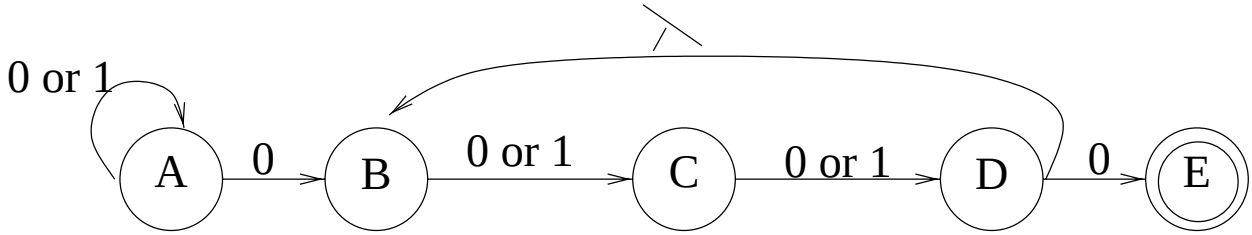
   (a) (5 points) Every LL(1) grammar is unambiguous. **true** ☐ **false** ☐ Explain why:

   (b) (5 points) Every Right-Linear grammar is an LL(1) grammar. **true** ☐ **false** ☐ Explain why:

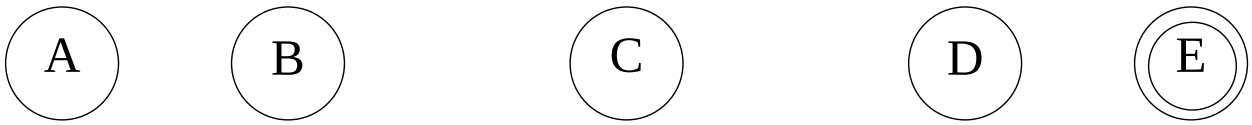
(c) (5 points) Every regular set has an unambiguous grammar. **true** ☐ **false** ☐
Explain why:

(d) (5 points) LL(1) parsers are more powerful than LR(1) parsers. **true** ☐ **false** ☐
Explain why:

4. (20 points) Consider the following FSA:



(a) (5 points) Below show the corresponding nondeterministic FSA without λ:



Continued on next page...

(b) (10 points) Below show your construction of the corresponding deterministic FSA (DFA):

(c) (5 points) Below show a regular expression for the language of the DFA you designed for question 4b. *Hint:* there is a clever way to do this easily.

5. (20 points) Design an LR(0) grammar for the language of problem 2 and show its LR(0) construction. You are welcome (indeed, encouraged) to use what you have installed from problem 2 to verify the LR(0)-ness of your solution.