UNIVERSITY OF CALIFORNIA
Department of Electrical Engineering
and Computer Sciences
Computer Science Division

Prof. R. Fateman

Fall, 2002: Sept 26,2002 3:30PM

CS 164 Midterm 1 PRINT YOUR LOGIN CS164-_____

Please read all instructions carefully.

There are 7 questions in the exam on 5 pages. Some questions have multiple parts.

You have 80 minutes to complete this test. The exam is closed book but you may refer to your one page (2-sides) handwritten 8.5 by 11 inch paper.

Please write your login name ON EVERY PAGE. Place the answers in the space provided. If you may use the backs of the exam pages for answers, clearly direct the grader to the location of answers. Solutions will be graded on correctness. Please write neatly. Partial credit will be given.

Your family name___________________________

Your first name ___________________________

Your LOGIN NAME **PRINTED IN CAPITAL LETTERS** CS164-______

Underline the name of your Teaching Assistant: Shoaib or Ryan

The DAY and TIME YOUR SECTION MEETS: ___________________________

READ AND SIGN THIS:

I certify that my answers to this exam are all my own work.

Signed: ____

question	grade	out of
1		25
2		4
3		5
4		6
5		5
6		10
7		15
total		70

1. [25 points] Here is a grammar for a simple language that includes array assignments like `id[id]= int+id`. We use the same notation for grammars as in lecture and assignments 3 and 4.

```
(defparameter
    G1
    '((S -> H $)          ;;rule 1
      (H -> lval = E)     ;;     2
      (E -> id)           ;;     3
      (E -> int)          ;;     4
      (E -> E T)          ;;     5
      (T -> + E)          ;;     6
      (T -> )             ;;     7
      (lval -> id)        ;;     8
      (lval -> id [ E ] );;     9
     ))
```

a. What is the set of terminal symbols in G1?

b. Is L(G1) finite or infinite? (Explain)

c. If x is an id, and 34 is an int, then which of these are sentences in L(G1)? Assume there is a $ following each string.

```
x[34]=x
x= x[34]
x[x[34]]
x= 34)
x=(x[34]+34)
x=x+x+x
```

d. Why is this grammar not suitable for LL(1) parsing?

e. Compute the needed First and Follow sets, and then draw the parsing table below.

	$	+	]	id	int	[
S						
H						
E						
T						
lval						

f. Write out a simple grammar for the same language that IS suitable for LL(1) parsing. We were able to do this by changing 3 rules, adding 2 rules and removing one rule.

2. [4 points] Define the following terms in complete English sentences.

a. Syntax directed parser generation

b. Context Free Grammar

c. Nondeterministic Finite Automaton. (Assume DFA is defined)

d. Leftmost derivation (Assume derivation is defined)

3. [5 points]

a. For each of the following languages $L1$, $L2$ and $L3$ write out all strings of length less than or equal to 4, starting with the shortest. Let the regular expression A denote the set {a}, and B denote {b}. .

$L1 = A|B^*$ contains:

$L2 = A^*B^*A^*$ contains:

$L3 = (A^*B^*)^* + L1 + L2$ contains:

b. You are told that for a particular regular expression `R` that `R` denotes the same set as the regular expression `RR`. What can you say about `R`?

c. An extra edge is added to an NFA recognizing the regular expression Q. The edge is an ϵ transition from the (single) accepting or final state to the (single) start state. What regular expression does the new NFA accept?

4. [6 points] Any finite state machine can be encoded in a context free grammar by using a certain pattern of rules, where each rule has a restricted form as in the example below:

```
(defparameter
    G2
    '((S -> a X)
      (X -> b X)
      (X -> c Y)
      (Y -> c)))
     ))
```

a. Draw a DFA accepting the same language as G2. Be sure to indicate start and final states.

b. Give an example of a grammar whose language *cannot* be recognized by a finite state machine. Use at most 2 rules.

5. [4 points] Here are the rules for a grammar G3 with start symbol `S`

$$S \to aSb$$

$$S \to c$$

Complete writing a recursive descent parsing program `parse` that returns `yes`, given a lisp list that constitutes a sentence in L(G3). We give you two useful parts already.

```
(defun parse (tokens)(s)(if (empty tokens) "yes"))

(defun eat(h) (cond((equal h (car tokens))(pop tokens))
                (t (error "stuck at ~s" tokens))))

;; sample test:  (parse '(a c b)) ;; acceptable
```

6. [10 points]

Write down the result of running your Tiger lexical analysis program `fsl` on a file containing only this material starting on line 1:

```
abc_def 123 "abd\    \ def" if /* "hello */ ( "world")+2 /*/*/ foo
```

7. [15 points]
Here is a simple grammar which is not LR(0). Prove this to be the case by completing the LR(0) diagram below, adding items, showing reductions and **drawing labelled edges between states**. Look carefully at state 3 to see the problem, and explain.

```
(defparameter gram4 '(
   (S ->  E $)    ;; rule 1
   (E ->  P x E)  ;;      2
   (E -> P)       ;;      3
   (P -> id)      ;;      4
     )

State 1:  S -> . E $


State 2: S -> E . $


State 3:  E -> P . x E


State 4: E -> P x . E


State 5: P -> id .


State 6: E -> P x E .
```