Midterm Exam
CS 314, Spring ’99
March 4, 1999

Name: ________________________________

SSN: ________________________________

Section: _____

Instructions

I have tried to provide enough information to allow you to answer each of the questions. If you need additional information, make a reasonable assumption, write down the assumption with your answer, and answer the question. There are five problems. Good luck!
Problem 1 – regular expressions and finite state automata (10 pts)

Typically, identifiers in programming languages are specified as regular expressions that can be recognized by finite state automata. Assume that you want to restrict the set of valid identifiers as follows:

1. Identifiers can only contain letters “a”, “b”, and digits “1”, or “2” as their symbols, i.e., are words over the alphabet { a, b, 1, 2 }.
2. Identifiers have to start with at least two letters, followed by any sequence of letters and digits, and have to end with the digit “1”.

For example, the words aa1 and ba22a1 are valid identifiers, but b2a1 and 1aaa1 are not.

1. Give a regular expression for this language of identifiers (4pts).

   Answer:

   \((a|b)(a|b)(a|b|1|2)^* 1\)

2. Give the transition diagram of a DFSA that recognizes the language of valid identifiers by adding only edges and edge labels in the partial transition diagram below. Adding additional states or changing the final and/or starting states is not allowed. NOTE: Your DFSA may contain states for which there are no transitions on some input symbols (6pts).

![Transition Diagram]

NOTE: state \(S_1\) is the start state, and state \(S_4\) is the final state.
Problem 2 – context-free and attribute grammars (10 pts)

1. Give a context-free grammar for the following language \( L \subseteq \{a, b, c, d\}^* \) (4 pts)
   \[ L = \{a^n c^m b^n d^m \mid n \geq 0, m > 0\} \]
   \[ Answer: \]
   \[ S ::= a \ S \ d | A \]
   \[ A ::= c \ A \ b | c \ b \]

2. Show a parse tree for the word \( aaebdd \in L \) for the grammar you defined above (3 pts).
   \[ Answer: \]
   \[
   \begin{array}{c}
   S \\
   / \backslash \\
   a S d \\
   / \backslash \\
   a S d \\
   / \backslash \\
   c b
   \end{array}
   \]

3. Prove that the following grammar is ambiguous. If you think that the grammar is not ambiguous, give a short argument. (3 pts)
   \[ S ::= A \ a \ B \]
   \[ A ::= a \ | b | \epsilon \]
   \[ B ::= a \ | c \ | \epsilon \]

   \[ Answer: \]
   The grammar is ambiguous since \( aa \) is a sentence that has two distinct parse trees:

   \[
   \begin{array}{c}
   S \\
   / \backslash \\
   A a B \\
   / \backslash \\
   \epsilon a \\
   a \ a
   \end{array}
   \]  
   \[
   \begin{array}{c}
   S \\
   / \backslash \\
   A a B \\
   / \backslash \\
   \epsilon a \\
   a \ e
   \end{array}
   \]

   Remark: \( e \) stands for \( \epsilon \).
Problem 3 – Unification (9 pts)

If the most general unifier (mgu) exists, write down the unifier. Otherwise answer no. (3 pts each)

• foo1(A,[b],C) and foo1(a,D).
  
  Answer:
  
  no

• [A,b,A] and [a | D].
  
  Answer:
  
  A = a
  D = [b,a]

• foo2(B, [B | C]) and foo2(A; [a | B]).
  
  Answer:
  
  B = C = A = a
Problem 4 – Prolog (12 pts)

1. Write Prolog predicates as described below, (3 pts each)
   - Determine the second-last element in a list of symbols. You can assume that the input list has at least two elements.

   Example:
   ```prolog
   ?- second-last(X, [a,b,c,d]).
   X = c;
   no
   
   Answer:
   second-last(X,[A,B]) :- X = A.
   second-last(X,[A|B]) :- second-last(X,B).
   ```

   - Take a list of integers as input and generate as output a list where each input value has been incremented by 1.

   Example:
   ```prolog
   ?- add1([1,3,2,5], X).
   X = [2,4,3,6];
   no
   
   Answer:
   add1([],[]).
   add1([X|XR],[Y|YR]) :- Y is X+1, add1(XR,YR).
   ```
• Remember our member predicate:

\[
\text{member}(A, [A \mid B]),  \\
\text{member}(A, [B \mid C]) \leftarrow \text{member}(A, C).
\]

Write a member1 predicate that does the same as the member predicate with the restriction that it only returns a single answer in the cases where member returns multiple answers.

Example:

?- member(X, [a,b,c]).
X = a ;
X = b ;
X = c ;
no

?- member1(X, [a,b,c]).
X = a ;
no

Answer:

\[
\text{member1}(A, [A \mid B]) \Leftarrow !,  \\
\text{member1}(A, [B \mid C]) \leftarrow \text{member1}(A, C).
\]

2. Draw a Prolog search tree (book version) for the following query based on your definition of predicate second-last above. Assume the user hits “;” as long as there are answers, i.e., show the entire search tree. (3 pts)

•  

Answer:

```
   second_last(X, [1,2,3])
   
   X->X
   A->1
   B->[2,3]

   second_last(X, [2,3])
   
   X->X  
   X->2
   A->2
   B->3  
   
   yes  
   second_last(X, [3])
   
   X->X
   A->3
   B->[ ]
   
   second_last(X, [ ]) 
   fail
```
Problem 5 – quick identifications (9 pts)

Answer the following questions with either yes or no by marking the appropriate box as your selected answer. Note: Do not give any justification for your answer! (1 pt each)

1. For every regular expression there is a left-linear context-free grammar that generates the same language?
   
   Answer: [ ] Yes [x] No

2. Every context-free grammar is also a context-sensitive grammar?
   
   Answer: [ ] Yes [x] No

3. The tokens generated by the scanner are the non-terminal symbols of the context-free grammar recognized by the parser?
   
   Answer: [ ] Yes [x] No

4. Compilers map programs into other programs?
   
   Answer: [ ] Yes [x] No

5. Type errors can only be detected at compile time?
   
   Answer: [x] Yes [ ] No

6. Structured programming allows arbitrary go tos within single-entry / single-exit program components?
   
   Answer: [x] Yes [ ] No

7. The Random Access Machine (RAM) is the semantic model for imperative languages?
   
   Answer: [x] Yes [ ] No

8. A deductive calculus that is sound, but not complete, is useless?
   
   Answer: [x] Yes [ ] No

9. Prolog programs will terminate for all possible inputs?
   
   Answer: [ ] Yes [x] No