Midterm Exam
CS 314, Fall '98
October 20, 1998

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 (12 pts)

1. In the following transition diagram, $S_1$ is the starting state and $S_5$ is the final state. Describe in English the language that the NFSA recognizes (3 pts).

![Transition Diagram]

Answer:

2. Why is the shown automaton a NFSA and not a DFSA (deterministic finite-state automaton) (3 pts)?

Answer:

3. Give a regular expression for the same language as recognized by the NFSA (3 pts).

Answer:

4. Give the transition diagram of a DFSA that recognizes the same language 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 (3 pts).

Answer:

![Transition Diagram]
Problem 2 – context-free and attribute grammars (9 pts)

1. Give a context-free grammar for the following language $L \subseteq \{a,b,c,d\}^*$. Use the BNF notation to specify your grammar (3 pts).

   $L = \{ a^n c^m b^n d^n \mid n > 0, m > 0, o \geq 0 \}$

   Answer:

2. The following grammar generates the language of matching parenthesis.

   $S ::= S \mid ( \ S \ ) \ | \ ( \$

   Prove that this grammar is ambiguous (3 pts).

   Answer:

3. Write an attribute grammar for the language of matching parenthesis that determines the maximum nesting depth of parenthesis (3 pts). Examples:

   - () has nesting depth 1.
   - (()) has nesting depth 2.

   Your attribute grammar has a single attribute, called depth. Specify the semantic actions that will result in the computation of the maximum nesting depth.

   Answer:

   \[
   S ::= S_1 \ S_2 \quad \{ \quad \}
   \]

   \[
   S ::= (S_1) \quad \{ \quad \}
   \]

   \[
   S ::= ( \quad ) \quad \{ \quad \}
   \]
Problem 3 – typing (6 pts)

Type equality (or type equivalence) can be defined in different ways, namely as structural, name, or declaration equivalence. The following program segment introduces three data objects \(x\), \(y\), and \(z\).

type string = array (1..10) of character;

type student = record name: array (1..10) of character;
    ID : integer;
end;

type professor = record name: string;
    ID : integer;
end;
x: record name: string;
    ID : integer;
end;
y: student;
z: professor;

Answer the questions whether two data objects are structural, name, or declaration equivalent with either yes or no by marking the appropriate box as your selected answer.

<table>
<thead>
<tr>
<th></th>
<th>structural?</th>
<th>name?</th>
<th>declaration?</th>
</tr>
</thead>
<tbody>
<tr>
<td>x and y</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>x and z</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>y and z</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
</tr>
</tbody>
</table>
Problem 4 – C++ (10 pts)

The following C++ code contains illegal statements.

1. Identify the illegal statements or functions by underlining them in the program listing.
2. Ignoring the statements or functions that are illegal, show the output of the program.

```cpp
#include <iostream>

// I/O example: cout << "hello " << me;
// will write the string "hello" followed by the value of variable
// "me" onto the standard output

class car {
    private:
        char *type;
    protected:
        char *model;
    public:
        car() {type = "A"; model = "UNKNOWN\n";} // constructor
        void virtual printModel() { cout << "in car: " << model; }
};

class chevi : public car {
    public:
        chevi() {model = "CHEVI\n";} // constructor
        void printType() { cout << "in chevi: " << type; }
        void printModel() { cout << "in chevi: " << model; }
};

void tip(car *a) { a->printType(); }
void top(car *a) { a->printModel(); }

main()
{
    car *a = new car;
    chevi *c = new chevi;

    cout << a->type;
    cout << c->model;
    tip(a);
    top(a);
    tip(c);
    top(c);
    c = a;
    top(c);
}
Problem 5 – quick identifications (10 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 finite-state automaton (FSA) that recognizes the same language?
   Answer: [ ] yes [ ] no

2. The number of transitions performed by a DFSA to accept a word consisting of n symbols is exactly n?
   Answer: [ ] yes [ ] no

3. Programs that do not contain any explicit memory deallocation statements such as delete will not have dangling pointers during program execution?
   Answer: [ ] yes [ ] no

4. Interpreters map programs into their answers?
   Answer: [ ] yes [ ] no

5. Garbage collection is a compile-time technique to reduce the memory requirements of compiler generated programs?
   Answer: [ ] yes [ ] no

6. The memory layout of a data object typically depends on its expected lifetime?
   Answer: [ ] yes [ ] no

7. Explicit dynamic memory allocation through operations such as new are typically done on the heap, and not on the stack?
   Answer: [ ] yes [ ] no

8. Function and operator overloading is a form of polymorphism?
   Answer: [ ] yes [ ] no

9. Strongly typed languages allow some type errors to remain undetected, but only in a very limited set of cases that don’t happen very often?
   Answer: [ ] yes [ ] no

10. The row-major order layout of multi-dimensional arrays is typically used in programming languages since column-major layouts have more complex address computations?
    Answer: [ ] yes [ ] no