CSCE 531 Fall 2001
SECOND MIDTERM EXAM
Friday 01/11/2—Two Pages, Closed Book, 60 points maximum

1 Short Questions–1 point each

1. BNF production rules may describe context-free languages. True or false?
2. BNF production rules and regular expressions have the same expressive power. True or false?

2 Lexical Analysis

(Parts of this question are from `http://www.cs.purdue.edu/homes/hosking/502/`) Certain assemblers form their integer literals in the following way. *Binary* literals consist of one or more binary digits (`0, 1`) followed by the letter `B`; e.g., `10110B`. *Octal* literals consist of one or more octal digits `0` through `7` followed by the letter `Q` (since `O` looks too much like `0`; e.g., `1234567Q`. *Hexadecimal* literals consist of at least one decimal digit (`0` through `9`) followed by zero or more hexadecimal digits (`0` through `9` and `A` through `F`) followed by the letter `H`; e.g., `0ABCDEFH`. *Decimal* literals consist of at least one decimal digit *optionally* followed by the letter `D`; e.g., 1234.

1. (15 points) Draw the state diagram of an NFA (not a DFA) for these literal forms; you may use ϵ-transitions.
2. (5 points) Give a regular expression for the literals; you may use ϵ.
3. (Bonus Question—10 points) Convert the NFA to a DFA. (Hint: From the initial state, consider the following three transitions: resulting from consuming `0` or `1`, resulting from consuming `2-7`, resulting from consuming `8` or `9`.)

3 Parsing

(Parts of this question are from `http://www.cs.purdue.edu/homes/hosking/502/`)
A sample variable declaration block of Pascal is `var foo, bar: int; x, y: real; c: char`
Consider the following grammar for Pascal variable declarations:

```
<var-decl> ::= var <decl-list>
<decl-list> ::= <decl> ; <decl-list> | <decl>
<decl> ::= <id-list> : id
<id-list> ::= <id-list>, id | id
```

1. (5 points) Transform this grammar (using left factorization or elimination of left recursion or both) in a form that can be parsed by an LL(1) parser.

2. (20 points) Design a recursive descent parser for the <var-decl> grammar. Do so by defining the class `Parser` containing the private member variable `currentToken`, the auxiliary methods `accept` and `acceptIt`, the parsing methods corresponding the production rules of the transformed grammar, and the method `parse`. Assume that the `scan` method is already written.

4 Contextual Analysis

1. (8 points) *Briefly* describe the difference between monolithic, flat, and nested block structures and explain how this affects management of the identification table.

2. (5 points) There are two common scope rules for the standard environment. (One is used in C.) Contrast them briefly.