Name: ______________________

CSci 401 Introduction to Compilers
Final Exam

Fall 1998 W.L. Ruzzo

Instructions: closed book, closed notes, 110 minutes, 200 points.
Don't open until instructed.

1	/15
2	/15
3	/15
4	/20
5	/20
6	/20
7	/15
8	/20
9	/30
10	/30
Total	/200

1. (15 points) Is it a good idea to have the scanner return a token to the parser for each comment? Why or why not?

2. (15 points) Show that the following grammar with start symbol S is ambiguous:

$$\begin{array}{rcl} S & \Rightarrow & A \mid B \\ A & \Rightarrow & bc \mid Ac \\ B & \Rightarrow & bB \mid cB \mid c \end{array}$$

3. (15 points) For a nonterminal X in a context-free grammar, what is FOLLOW(X) intended to summarize?

4. (20 points) What is an *activation record*? When is it built? When is it destroyed? What information is stored in it?

5. (20 points) Under what circumstances would space for a variable not be allocated on the run time stack?

6. (20 points) What is a *display*? True or false: in a language like PL/0, displays cannot be statically allocated, because their maximum sizes are not known at compile time. Justify your answer.

7. (15 points) Consider the following program:

```
Module M;
  var x:int;

  procedure bar(y:int);
  begin
    y := x+1;
    y := x+1;
  end bar;

begin
  x := 0;
  bar(x);
  output := x;
end M.
```

(a) What are the possible outputs of this program assuming *call-by-value* parameter passing?

(b) What are the possible outputs of this program assuming *call-by-reference* parameter passing?

(c) What are the possible outputs of this program assuming *call-by-value-result* (also called *copy in/copy out*) parameter passing?

8. (20 points)

(a) What is a *basic block*?

(b) Why is it harder to do certain optimizations, constant propagation for example, *across* basic block boundaries than it is to do them *within* one basic block?

9. (30 points) Assume we have declared

```
var
  row:int;
  col:int;
  x: array[100] of array[5] of int;
```

Sketch the abstract syntax tree corresponding to the statement

```
x[row][col] := 42;
```

Sketch the intermediate code that would be generated from this statement, and briefly outline the processing that would go on during IR generation as this portion of the AST is processed.

10. (30 points) Suppose we want to add **goto**'s to PL/0: Any statement may be labeled by preceding it by an identifier and a colon, and control is transferred to that statement by a **goto** statement naming that label. I.e., we add rules

    Stmt ::= **Id :** Stmt | **goto Id** | ...

    to the grammar.

    (a) What changes have to be made to the scanner to handle this?

    (b) What changes have to be made to the parser to handle this?

    (c) What changes have to be made to semantic analysis to handle this? For example, do goto's into compound statements from outside make sense? Goto's into if statements? Loops? (Assume that the identifiers in statement labels and goto's follow the usual scope rules.)

(d) What changes have to be made to code generation to handle *local* goto's, i.e., ones whose target is in the same procedure?

(e) What changes have to be made to handle *non-local goto's*, i.e., those whose target is in a different (but lexically visible) procedure? Don't forget to consider what happens to the stack, saved registers, etc. and how you can get the information you need to do those changes at runtime.