EE573 Fall 98, Midterm Exam

Name _____

Hints:
- The available space indicates the approximate, expected length for your answer.
- Always add an explanation for all yes/no answers.

1. (5 pts) Explain why a compiler might use two internal representations, such as a syntax tree and a 3-address code form. Give an example of the use of each representation.

2. (10 pts) Give two examples of the static semantics of a program and explain where in the compilation process this information is used.

3. (14 pts) Draw a deterministic finite automaton and the transition table for the regular expression [ab[c]+]*d
[]+ stands for 1 or more repetitions
[]* stands for 0 or more repetitions

4. (7 pts) What is the name of the following transformation. Is this transformations always beneficial?

```
-- original code:
  do i=1,n
    a(i*const)= expr
  enddo

-- transformed code:
  i0 = 0
  do i=1,n
    i0=i0+const
    a(i0)= expr
  enddo
```

5. Consider the following program fragment:

```
I1:  a = b + c*d
     ...
          call xyz(d)
          ...
I2:  e = c*d
```

a) (8 pts) Common subexpression elimination tries to determine that the term c*d does not need to be recomputed for statement I2. Explain what information the compiler needs to keep about variables and temporaries in order to make this determination? Describe how this information is maintained/updated as the optimizer traverses the program.

b) (4 pts) In the above example, variable d is passed by value to subroutine xyz. Does this information help the optimizer? Why or why not?

6. (4 pts) Would you call the following optimization a peephole optimization? Why or why not?

```
add variableX,R3
store R3 variableY    -->  add variableX,R3
```

(Explanation: the store operation could be eliminated because variableY is not re-used in the program.)

7. A programmer says he optimized a program by replacing a subroutine-local array with a statically-allocated, global array, as in the following example:

-- original program:

```
int main()
        {  double a[1000][1000];
           int i,j;
           ...
           for (i=0;i<1000;i++)
             for (j=0;j<1000;i++)
                     a[i][j]= some expression
           ...
        }
```

-- transformed program:

```
double a[1000][1000];
int main()
        { int i,j;
           ...
           for (i=0;i<1000;i++)
             for (j=0;j<1000;i++)
                     a[i][j]= some expression
           ...
        }
```

(12 pts) Explain how this transformation can affect the program performance. Describe compiler optimizations that may be relevant for this discussion.

8. (6 pts) Name three (non-trivial) implementation issues for subroutine inline expansion.

9. a) (10 pts) For the switch statement of the C programming language, write a grammar and annotate it with semantic action symbols (the default clause can be omitted).

b) (8 pts) Describe the information that needs to be communicated between the action routines. For each information item, indicate the producer and the consumer actions.

10. (8 pts) Explain how you would classify a parser that operates as follows:

Repeat
1: scan an input token and push it onto the stack
2: if the top n stack symbols represent the RHS of a production, replace them with the production's LHS symbol.

11. (4 pts) Can the following LL(k) grammer be rewritten as a LL(k-1) grammar? If yes, do so. If no, explain why. What is k in this example?

start -> A B C
A -> c d e
B -> c d f
C -> c d g