Name: _____

EE573 Midterm Exam, Fall 2000

Please write your answer in the provided space. If you think a question is ambiguous, state your assumptions and solve the problem under these assumptions.

1. (2pt) Compilers have an analysis part and a synthesis part. Name three compiler techniques each that belong to the analysis part and to the synthesis part, respectively.

Analysis: 1_____ 2_____ 3_____

Synthesis 1_____ 2_____3_____

2. (2pt) Scanners and parsers can be generated automatically because (mark all correct reasons)

O they are the least complex part of the compiler
O their functionality can be specified with formal methods
O it is easy to define the semantics of a programming language
O of the extended BNF form
O regular expression can be translated into finite automata

3. (2pt) Name three intermediate representations used by compilers

1_____ 2_____ 3_____

4. (2pt) Name four attributes of symbol table entries

1_____ 2_____ 3_____ 4_____

5. (4pt) In the statement *IF <expr> THEN stmt ENDIF*
Explain the syntax, static semantics, and execution semantics

Syntax: _____

Static semantics: _____

Execution Semantics: _____

6. (2pt) What is the most frequently used method for specifying the semantics of a programming language?

7. (4pt) Give an example of how the evolution of compilers has changed computer architectures.

8. (8pt) Create a DFA for the regular expression a (b c+)*. Write the transition table.

9. (8pt) Given the terminal symbols A, B,C, and D, write a grammar that can generate exactly the following four programs.
A B C, A A B, A C B, A B C B

10. (6pt) Consider the production
dostmt → DO id = expr , expr #begindo stmt ENDDO #enddo

Explain a basic problem with handling these semantic actions in an LR parser. Show how the problem can be resolved?

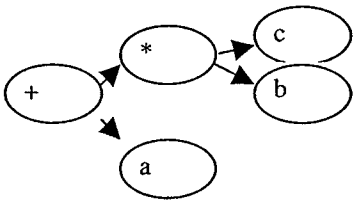
11. (8pt) Given a parse stack of a shift-reduce parser that contains the actual symbols (rather than states). In the following grammar, how does the parse stack look like after scanning the input up to symbol V (including V)? Hint: the first element is the symbol A.

Input : A B C D X Y Z U V P Q R

Grammar:
goal → A B N4 Q R
N4 → N1 Y N3 V P
N1 → C D X
N3 → Z U

Answer: A

12. (12pt) Write semantic action procedures for #term and #ttail such that they help create the shown syntax tree rather than generating code for the expression a+b*c. Refer to semantic records of the RHS symbols as $1, $2, etc., and $$ for the LHS. Use the notation $1.field for accessing a specific field in the semantic record. Write only the essentials of the procedures.



EXPR = TERM ETAIL
ETAIL → + TERM ETAIL
ETAIL → λ
TERM = FACTOR TTAIL #term
TTAIL → * FACTOR TTAIL #ttail
TTAIL → λ
FACTOR = ID | literal

13. (6pt) The code generated for accessing a value parameter in a subroutine is something like this:
load fp(5) r0, where *fp(5)* means "offset 5 relative to the frame pointer".
What code would you generate to load the value of a reference parameter?

14. (4pt) Give two reasons why and in what situations saving and restoring registers at subroutine calls can cause significant execution overhead.

1:_____

2:_____

15. (6pt) Write a program example that will require static links in the generated stack frame. Explain what it is in your program that uses the static links.

16. (6pt) For CSE professor May B. Right developed a technique to reorder an expression a+b+c so that it can be recognized to match a previous expression c+b+a in the program. Can you think of a reason why this technique may cause incorrect execution results?

17. (10pt) Many compilers use "name-only analysis" for detecting aliases. Give a high-level description of a simple test that would correctly detect the precise aliases in the following code example.

```
1: A(i,j)   := A(i,j)+B+C;
2: A(i,j+1) := A(i,j)+B+D;
3: A(i,j)   := A(i,j)+B;
```

18. (8pt) In the bottom up register allocation algorithm discussed in class (slide 167) , registers that are no more used are freed immediately. If you removed the two lines that free the registers r_x and r_y , how would you modify the allocate function to achieve the same (or a simlar) effect?