Name_____

EE573 or EE468 468 Seat #56 96/90 High

Exam #1
R.Eigenmann
Fall 2001

***** FORM A ***** FORM A ***** FORM A ***** FORM A *****

Circle all answers that apply, some questions may have multiple answers.
If you think a question is ambiguous, state your assumptions.
Advanced questions are bonus questions for EE468 students.

1. From the syllabus, the following is considered cheating:
(a) copying from a neighbor during an exam
(b) copying a program from somebody else (not in my group) for my project
(c) giving my project to a class mate so she can learn what I did
d- inadequately protecting my files so somebody else can copy from me.
(e) using (part of) a project that I happened to find on some computer

2. The following are tasks of a compiler
(a) translate a source program into assembly code
(b) translate assembly code into machine code
c- execute a program
(d) translate source code into modified source code
e- interpret a program

3. Compiler passes do the following:
(a) scanners tokenize the input character stream
(b) parsers translate a grammar into syntactic constructs
c- semantic routines read the scanned tokens and generate 3-address code
(d) optimizers passes improve the code by some measure
(e) code generators issue assembly code

4. Automatic generation of compiler passes:
(a) scanners can be generated with automatic tools
(b) parsers can be generated automatically from grammar specifications
c- semantic routines are difficult to automate because grammars cannot be specified precisely
(d) automating the generation of optimizers is still a research topic
e- code generators can be created automatically from the precise specifications of machine instructions

5. Internal representation:
a- the fewer compiler passes there are, the more internal representations are needed
b- the syntax tree representation is typically generated by the parser
c- 3-address code is a more architecture-independent representation than assembly code
d- compilers have exactly three internal representations: tokens, syntax tree, and 3-address code
e- the term "internal representation" means all data structures used by a compiler.

6. Syntax and semantics: Consider the phrase FOR i=1 TO N DO STATEMENTS
a- the syntax of this phrase has six tokens
b- the semantics describes the effect that this phrase has
c- the syntax describes the grouping of characters into tokens
d- the semantics can be derived from a well-defined syntax
e- the context-free grammar may specify that N must be previously declared

7. Compilers, languages, architectures:
a- advances in computer architectures necessitate advances in programming languages
b- advances in computer architectures necessitate advances in compiler technology
c- the rise of high-level languages has simplified compiler design
d- better defined languages tend to lead to smaller grammar specifications
e- even if the ISA of a processor is the same, a new processor may lead to changes in its compiler, because code generation alternatives may now perform differently.

8. MICRO compiler:
a- EE468EXAM1 would be a correct identifier for MICRO
b- 573.468 would be a correct literal for MICRO
c- the MICRO parser can always tell which of the three alternatives of the production for "statement" to choose by looking at the next token
d- in MICRO, operator precedence is in the order "PLUSOP" then "MINUSOP"
e- an empty statement (two semicolons in a row) is legal syntax in MICRO

9. Recursive descent parsing:
a- every terminal symbol has an associated parse procedure
b- productions that contain alternatives lead to parse procedures with some form of conditional execution
c- if the parse procedure cannot determine which production to take next by looking at the next token, then the current production is empty.
d- the parse procedure of the production L :== A | B , where A and B are terminals, may look like this: procedure L() {if (Lookahead(A)) match(A) else match(B) }
e- the above procedure is wrong because we also have to lookahead for B

Not "Semantic processing in Micro"

10. Semantic processing:
(a) semantic annotations are grammar extensions that cause the parser to call semantic action procedures.
(b) semantic records are data structures through which semantic action procedures communicate with each other
(c) not every production needs to have a semantic annotation
(d) semantic records may contain information about parsed symbols as well as results of code generation actions.
(e) designing and implementing action procedures is one of the most time-consuming tasks of a compiler design.

for Micro this is Not true

11. Scanners and finite automata:
(a) a scanner can be implemented using finite automata
b- all finite automata can be expressed through transition tables
(c) an error state in a transition table corresponds to an incorrect program input
(d) regular expressions can be translated automatically into DFAs
(e) the two dimensions of a transition table are the states of the DFA and the range of legal program input characters

12. Value of tokens:
(a) scanners must recognize legal tokens AND their values
b- all tokens have a value
c- the value of an identifier is its length
(d) transducers are helpful in deriving the value of a token
(e) filter functions in Lex help identify the token value

13. BNF and Grammar:
(a) Extended BNF can be translated into standard BNF
(b) Extended BNF is more concise
(c) in standard BNF a Nonterminal can have several productions
(d) Given a Nonterminal and a lookahead token, Parsers for LL(1) grammars can determine which production to take by looking at the First Sets.
e- the following production may be part on an LL(1) grammar: Expr ::= Expr ExprTail

No, this is left-recursive

14. Parsing:
(a) the parse table indicates which production to take, given a Nonterminal and a lookahead character
(b) Recursive descent parsers can be generated automatically from parse tables.
(c) Parse tables can be interpreted directly by a stack-based parser driver
d- Maybe (c) is true, but semantic actions cannot be called by such a driver
e- All grammars can be turned into LL(1) but they may get very large

ADVANCED QUESTIONS:
(Bonus for EE468 students)

15. The following non-LL(1) grammars elements can be turned into LL(1):
(a) productions with a common prefix
(b) left recursions
(c) the "if-then-else" problem
d- all LL(k) grammars
e- all grammars than can be parsed with LR parsers

16. Is the following grammar LL(1)?
P ::= A B C D E F
A ::= a
B ::= b
C ::= c
D ::= d
E ::= e
F ::= f
a- yes, because you need to look at the first sets to decide which production to take.
b- no, because the firsts sets are irrelevant in this grammar
(c) yes, the productions can be decided without looking at the first sets
d- no, because you need to look ahead by five tokens
e- this is an illegal grammar

17. Sequence of compiler passes:
(a) several passes can be combined into one
b- it is possible to create one-pass compilers for all major programming languages
c- the order of compiler passes must always be fixed
(d) if a compiler contains several optimization passes, their relative order may sometimes be reversed
e- assembly code generation is the last pass of any compiler

+15