Mark Scored:

KATHMANDU UNIVERSITY
End Semester Examination
August/September, 2017

Level : B. Sc. Course : COMP 409
Year : III Semester : I
Exam Roll No. : Time: 30 min F. M. : 10

Registration No.: Date :

SECTION "A"
[20 Q × 0.5 = 10 marks]

Encircle the alphabetical letter of the most appropriate answer.

1. In a compiler, keyword of a language are recognized during
[A] Parsing of a program [B] The code generation phase
[C] Lexical analysis of the program [D] Data follow analysis

2. The best data structure to check whether an arithmetic expression has balanced parenthesis is:
[A] List [B] Queue [C] Stack [D] Tree

3. Assume that the operators +, - , x are left associative and ^ is right associative. The order of precedence (from highest to lowest) is ^, x, +, - . The postfix expression corresponding to the infix expression a + b x c – d ^ e ^ f is
[A] - + a x b c ^ ^ d e f [B] a b c x + d e ^ f ^ -
[C]a b + c x d – e ^ f ^ [D] a b c x + d e f ^ ^ -

4. The analysis phase of compilation is to:
[A] Break up source program into pieces and impose a grammatical structures
[B] Creates IR of source program
[C] Determines the operations and records them in a structure
[D] All of the above

5. Which of the following is not performed by a preprocessor?
[A] File Inclusion [B] Macro processing
[C] Error Handling [D] Language extension

6. Which phase of compilation can find out that the upcoming sequence of numeric characters represents no number in the source language?
[A] Code optimization [B] Lexical Analysis
[C] Semantic Analysis [D] Syntax Analysis

7. For a NFA of n states, corresponding DFA can have
[A] n^2 states [B] 2^n states [C] 2^{n-1} states [D] n/n^2 states

8. For the given ε-NFA, the equivalent DFA will have which of the following optimal states?

δ	ε	0	1
→q0	Φ	q1	Φ
q1	q0	Φ	q2
*q2	Φ	Φ	q0

[A] [q0], [q0,q1], [q2] [B] [q0,q1], [q1], [q1,q2]
[C] [q0], [q1], [q1,q2] [D] [q0], [q1], [q2]

9. A set of strings in the input for which the same token is produced as output is called?
[A] token [B] pattern [C] lexeme [D] parse tree

10. Which of the following is not a role of a parser?
[A] It obtains a string of tokens for a lexical analyzer
[B] It should report any syntax error in the program
[C] It should keep tokens in the symbol table
[D] It should recover from the errors so that it can continue to process the rest of the input

11. Drawback of recursive descent parser is:
[A] Suffers with left recursion
[B] Parse with top down method
[C] Recursion is not an issue
[D] Each non terminal is implemented as a procedure of recursion

12. If n is a cat-node with left child C1 and right child C2 and i is a position in lastpos(C1) then,
[A] all position in lastpos(C2) are in followpos(i)
[B] all position in firstpos(C1) are in followpos(i)
[C] all position in firstpos(C2) are in followpos(i)
[D] all position in lastpos(C1) are in followpos(i)

Consider the following grammar
S→aBDh
B→cC
C→bC | ε
D→ EF
E→ g | ε
F→f | ε
Question (13 to 14) are based on the given Grammar:

13. FOLLOW(B) is:
[A] {g,f,h} [B] {g,h} [C] {f,h} [D] {a,g,h}

14. FOLLOW(F) is equivalent to:
[A] FOLLOW(B) [B] FOLLOW(C) [C] FOLLOW(E) [D] FOLLOW(D)

15. Which of the following is not true for SLR?
[A] Parser is largest in size [B] It requires less time and space
[C] It exposes less syntactic features [D] Immediate error detection is not done

16. Consider the grammar G: { S →iEtS | iEtSeS | a ; E→b }
The Left factored grammar is:
[A]{ S→iEtSS' | a ; S' →eS | ε ; E → b} [B] { S→iEtS' | a ; S' →SeS | ε ; E → b}
[C]{ S→iS' | a ; S' →EtSeS | ε ; E →b} [D] { S →iES' | a ; S' →tSeS | ε ; E→ b}

17. For the grammar G: { E → E + T ; E →T; T→ T*F; T→F; F→ id} the number of states for SLR automation is
[A] 7 [B] 8 [C] 9 [D] 10

18. For grammar in Q17, Goto [I0, +] is :
[A] {T→T*.F; F→id} [B] {E→ E+.T; T→.T*F; T→.F; F→.id}
[C] {T→T*F.} [D] {E→ E.+T; E→ .T}

19. if L<R gotoYesTarget
gotoNoTarget
Which of the following can use the given code?
[A] While condition [B] Procedure call
[C] Function call [D] Pointer assignment

20. How many states are required to validate keyword *printf*
[A] 6 [B] 7 [C] 8 [D] 5

KATHMANDU UNIVERSITY
End Semester Examination
August/September, 2017

Level : B. Sc. Course : COMP 409
Year : III Semester : I
Time : 2 hrs. 30 mins. F. M. : 40

SECTION "B"
[6Q. ×4 = 24 marks]

Attempt *ANY SIX* questions.

1. Explain with specific example how tokens are specified. Also explain with subroutine how recognition of token is carried out during lexical analysis phase.

2. Construct DFA from the given regular expression over the alphabet {a.b}
R.E :**(aa|bb) (a|b)*aba (a|bb)**

3. Consider the following piece of code:

```
c = 10;
a = c;
i = j = 0;
while(i != (a*3))
{
        if (j) print("Hello");
        i++;
        d[i] = 10*i;
        b = 5 * i;
}
```

Identify different kinds of optimizations possible. Rewrite the code after making optimized.

4. Write a C-program that identifies the greatest number from n natural numbers. Convert your code into three address code.

5. Explain different error handling technique used in analysis phase of compilation.

6. Explain the significance of Intermediate Representation in compilation process.

7. Write Short notes on with example
   a. Local and Loop optimization
   b. Induction variable elimination
   c. Dead code elimination
   d. Constant folding and combining

SECTION "C"

[2 Q × 8 = 16 marks]

Attempt *ANY TWO* questions.

8. Consider the following grammar:
   S →aAC | bB
   A →Abc | Abd | e
   B → f | g
   C → h | i
   a. Design LL(1) parsing table for the grammar.
   b. Check whether the string w = **aebch** can be parsed or not.

9. Construct the following grammar:
   E → E+T | T
   T → TF | F
   F → F* | a | b
   a. Construct LR(0) item sets
   b. Construct LR(0) parsing table
   c. Check whether the string w = **a+a+ba*** can be parsed or not.

10. Consider the following grammar and give the syntax directed definition to construct parse tree. For the input expression **5*3+4*9**, construct an annotated parse tree along with dependency graph according to your syntax directed definition. (4+4)
    E → TE'
    E' → +TE'
    E' → ε
    T → FT'
    T' → *FT'
    T' → ε
    F → digit