August 2021

SYSTEM PROGRAMMING & COMPILER DESIGN

Time Allowed: 1.5 Hours | Full Marks: 70

Answer to Question No. 1 is compulsory and Answer any two questions from the rest.

1. Answer any ten of the following questions : 10x4

   i. What is ambiguity in CFG?
   ii. What is ϵ-closure for an ϵ-NDFA?
   iii. What is backpatching?
   iv. What is an LL(1) grammer?
   v. What is an item for an LR(0) parser?
   vi. What is handle pruning?
   vii. What is relocation?
   viii. Name the types of LR parser in the increasing order of their power.
   ix. What is an augmented grammer?
   x. What are quadruples?
   xi. What is loop optimization?
   xii. What is token attribute?
   xiii. What is lookahead pointer?
   xiv. What is left factoring?
   xv. What is a direct linking loader?
   xvi. What is meant by macro expansion?

2. a) Discuss briefly about the structure of a compiler touching upon all its phases with a neat block diagram.
   b) Explain how the following assignment statement would be translated through the different phases of compilation:
   position = initial + rate 7+8 60

3. a) List out the lexemes and their corresponding tokens for the C-code fragment below :

```
while ( count <= 10)
  {
    sum + = arr [ count ++ ] ;
  }
```

   b) Give the transition diagram for the lexical analyzer module that generates valid identifier tokens defined to be a string starting with a letter, followed by any number of letters or digits, and ending with a delimiter character.
   c) Explain briefly the input buffering scheme used by the lexical analyzer. (3x2)+5+4

4. a) (i) Give the ϵ-NDFA for the regular expression (a|b)*abb.
   (ii) Convert the ϵ-NDFA constructed in (i) into an equivalent DFA using *'subset construction method'*.
   (iii) Minimize the number of states of the DFA obtained in (ii).
   b) Considering the grammer below, draw the parse tree for the string
   "{ a = 1 ; { b = 0 ; } ; }" :

```
E -> I = C | { L }
L -> E ; L | ϵ
I -> a | b
C -> 0 | 1
```

   (3+6+3)+3

5. a) Disambiguate the following grammer that would generate arithmetic expressions involving operators '+', '–' (binary), '*', '/', '↑' and '–' (unary), denoting addition, subtraction, multiplication, division, exponentiation and negation respectively:

E -> E + E | E – E | E * E | E / E | E ↑ E | – E | (E) | a | b

b) Eliminate the non-immediate left recursion from the grammer below:

S -> Aa | b

A -> Ac | Sd | e

c) By left factoring, eliminate the non-determinism from the grammer below:

S -> aSSbS | aSaSb | abb | b

d) Discuss briefly about the working of a backtracking top-down parser with a suitable example.

4+3+4+4

6. a) Write recursive routines for a recursive-descent parser parsing for the grammer below:

E -> TE′

E′ -> +TE′ | ϵ

T -> (E) | a

b) (i) What is a handle?

(ii) Given the grammer below, find the handles of the right sentential forms of the reduction sequence for the input string *"id + id id $"* :

E -> E + E | E * E | id

(iii) Step through the sequence of moves a general bottom-up parser would make in parsing the input string *"id + id id $"* according to the grammer in (ii). 4+(3+4+4)

7. a) Check with proper justification whether the following grammer is LL(1) or not:

S -> iEtSS′ | a

S′ -> eS | ϵ

E -> b

b) Define operator grammer with an example. Convert the following non-operator grammer into an equivalent operator grammer :

S -> SAS | a

A -> bSb | b

c) (i) Construct the operator precedence table for the following operatorgrammer for generating arithmetic expressions involving operators '+' and '*', denoting addition and multiplication respectively:

E -> E + E | E * E | id

(ii) Using the table obtained in (i), show the sequence of moves an operator precedence parser would make on the input string *"id + id id $"*. https://www.wbscteonline.com 3+(3+2)+(3+4)

8. a) Consider the LL(1) grammer below:

E -> TE′

E′ -> +TE′ | ϵ

T -> FT′

T′ -> *FT′ | ϵ

F -> (E) | id

(i) Compute the FIRST and FOLLLOW sets for each non-terminal of the above grammer.

(ii) Construct the predictive parsing table for the above grammer.

b) Explain the action of the predictive parser of (a) by describing its sequence of moves as it attempts to parse the input string *"id + id id $"*. (6+4)+5

9. a) Discuss briefly about the components and working of an LR parser with a neat diagram.

b) Consider the augmented grammer below:

E′ -> E

E -> E + T | T

T -> T * F | F

F -> (E) | id

Find:

(i) CLOSURE ({ [E′ -> •E] })

(ii) GOTO ({ [E′ -> E•], [E -> E• + T] }, +) 9+(3x2)

10. a) Explain the concept of *'macro call within macro'* with an example.
    b) Discuss briefly about the two pass macro processor algorithm. 5+10

11. Write short notes on any three of the following: 3x5
    a) Symbol table
    b) Three address code
    c) 'Compile & go' loader
    d) Viable prefix
    e) Peephole optimization
    f) DAG representation
    g) SLR(1) parser
    h) Syntax directed translation
