B.TECH/CSE/7TH SEM/CSEN 4101/2017

COMPILER CONSTRUCTION
(CSEN 4101)

Time Allotted : 3 hrs **Full Marks : 70**

Figures out of the right margin indicate full marks.

Candidates are required to answer Group A and
<u>any 5 (five)</u> from Group B to E, taking <u>at least one</u> from each group.

Candidates are required to give answer in their own words as far as practicable.

Group – A
(Multiple Choice Type Questions)

1. Choose the correct alternative for the following: **10 × 1 = 10**

(i) Which one of the following statement is false for the SLR (1) and LALR (1) parsing tables for a context free grammar?
(a) The reduce entries in both the tables may be different
(b) The error entries in both the tables may be different
(c) The go to part of both tables may be different
(d) The shift entries in both the tables may be identical.

(ii) The language produced by the regular grammar S → aS|bS|a|b
(a) a*b* (b) aa*bb*
(c) (a+b)* (d) (a + b)(a + b)*

(iii) A given grammer is not LL(1) if the parsing table of a grammer may contain
(a) any blank field
(b) any Є-entry
(c) duplicate entry of same production
(d) more than one production rule.

(iv) Given the grammar S → ABc A → a|Є B → b|Є.FOLLOW(A) is the set
(a) {$} (b) {b} (c) {b, c} (d) {a, b, c}.

(v) We have the grammar E->E + n | E * n | n. The handles in the right-sentential form of the reduction for a sentence n + n * n are
(a) n, n + n and n + n * n (b) n, E + n and E * n
(c) n, E + n and E + E * n (d) n, E + n and E + n * n

(vi) The grammar S → Sa1|Sa2|b1|b2
(a) is left recurssive
(b) is a CFG
(c) has common left factor(s)
(d) is left recurssive and also has common left factor(s)

(vii) Peephole optimization
(a) is applied to a small part of the code
(b) can be used to optimize intermediate code
(c) can be applied to a portion of the code that is not contiguous
(d) all of these.

(viii) Optimization(s) connected with x : = x + 0 is(are)
(a) peephole and algebraic
(b) reduction in strength and algebraic
(c) peephole only
(d) loop and peephole.

(ix) What is used to keep track of currently active activations?
(a) Control stack (b) Activation
(c) Execution (d) Symbol.

(x) {S, A, B} is the non-terminal alphabet and {a, b} is the terminal alphabet of the CFG. S is the start symbol. The set of production rules are given below
S → aB S → bA B → b A → a B → bS A → aS B → aB B A → bAA
Which string is accepted by the given grammar?
(a) aabbbb (b) abbbba
(c) aabbab (d) aaaabb.

Group – B

2. (a) Convert the following regular expression to an equivalent NFA
(011 + 101 + 110 + 111)*

(b) Construct DFA directly (not by generating NFA) for the regular expression **(a | b) * ab**.

(c) List the tokens (with 'type'-s and 'value'-s) for:

```
int max (int x, int y)
    {
        Return (x > y? x : y);
    }
```

3 + 4 + 5 = 12

3. (a) Explain the different phases of a compiler, showing the output of each phase, using the example of the following statement:

```
for (i = 0; i < 10; i++)
    a = a + 10;
```

(b) What is the front end and back end of a compiler?

10 + 2 = 12

Group – C

4. (a) What language does the following CFG generate?

S → aB | bA
A → a | aS | bAA
B → b | bS | aBB

(b) Consider the following grammar and construct the SLR parsing table

E → E + T | T
T → TF | F
F → (E) | id

4 + 8 = 12

5. (a) Eliminate left recursion from the following grammar:

S → Ab | b
A → Ac| Sd | Є

(b) Find FIRST and FOLLOW sets of each nonterminal of the grammar obtained after solving part (a) of the question.

(c) Explain the terms "shift-reduce" conflict and "reduce-reduce" conflict in the context of LR parsers.

4 + 4 + 4 = 12

Group – D

6. (a) Define inherited and synthesized attributes with suitable examples.

(b)

$E \rightarrow E_1 * T$	{E.val = E_1.val * T.val}
$E \rightarrow T$	{E.val = T.val}
$T \rightarrow T_1 - F$	{T.val = T_1.val – F.val}
$T \rightarrow F$	{T.val = F.val}
$F \rightarrow 2$	{F.val = 2}
$F \rightarrow 4$	{F.val = 4}

Using the above Syntax Directed Translations, construct a parse tree for the expression 4 – 2 – 4 * 2 and show the evaluation order.

(3 + 3) + 6 = 12

7. (a) Construct a DAG and write the sequence of instructions for the expression

p + q / (r – s) + (r – s) * q.

(b) Generate three-address code for the above expression and implement it in quadruples, triples, and indirect triples.

6 + 6 = 12

Group – E

8. (a) Translate the following code it into machine code and show the register and address descriptors while the instructions are generated. (Assume that two registers are available.)

D : = B - C
E : = A - B
B : = B + C
A : = E - D

(b) Explain following terminologies:
(i) Absolute machine code
(ii) Relocatable machine code
(iii) Assembly code.

9 + 3 = 12

9. (a) Explain the following techniques of code optimization with suitable examples:
(i) Copy propagation
(ii) Dead code elimination
(iii) Code motion
(iv) Common sub-expression elimination

(b) Divide the following code into basic blocks and draw the flow graph.
(i) f =1;
(ii) i = 2;
(iii) if (i > x) goto (viii)
(iv) f = f * i;
(v) t = i+1;
(vi) i = t;
(vii) goto (iii)
(viii) goto calling program

(4 × 2) + (2 + 2) = 12