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 any 5 (five) from Group B to E, taking at least one 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) Consider the grammar with the following translation rules and E as the start symbol.
A→A1 # B {A.value = A1.value * B.value} | B {A.value = B.value}
B→B1 & C {B.value = B1.value + C.value} | C {B.value = C.value}
C→num{C.value=num.value}
What is the E.value for the root of the parse tree for the expression:
2 # 3 & 5 # 6 &4?
(a) 41 (b) 80 (c) 160 (d) 40

(ii) We have the grammar EXPR→ EXPR + id | EXPR × id | id. The handles in the right-sentential form of the reduction for a sentence id + id × id are
(a) id, EXPR + id and EXPR + EXPR × id
(b) id, EXPR + id and EXPR × id
(c) id, id + id and id + id × id
(d) id, EXPR + id and EXPR + id × id.

(iii) (a) An SDT consisting only of synthesized attributes is L-attributed.
(b) An L-attributed SDT can consist only of inherited attributes.
(c) An S-attributed SDT can have both synthesized as well as inherited attributes.
(d) If an L-attributed SDT has synthesized attributes, then it cannot have inherited attributes.

(iv) A system program that brings together separately compiled modules of a program into a form of language that is suitable for execution is called
(a) assembler (b) linking loader
(c) cross compiler (d) none of the above.

(v) Consider the expression below:
a<= –10 || a>= 10 **?** 100 **:** a*a. How many distinct tokens does it contain?
(a) 9 (b) 10 (c) 12 (d) 13.

(vi) Which of the following is not a content of activation record?
(a) Temporary variables (b) Return values
(c) Local variables (d) Address descriptor.

(vii) In some programming language, L denotes the set of letters and D denotes the set of digits. An identifier is permitted to be a letter followed by any number of letters or digits. The expression that defines an identifier is
(a) (L.D)* (b) (L + D)* (c) L(L.D)* (d) L(L + D)*.

(viii) Which one of the following statement is true?
(a) Canonical LR parser is more powerful than LALR parser
(b) SLR parser is more powerful than LALR
(c) LALR parser is more powerful than canonical LR parser
(d) SLR parser, canonical LR parser and LALR parser all have the same power.

(ix) There is a production rule in a grammar G:
$S \rightarrow a^k b \mid a^k c$
Here a^k means 'a' repeated k times.
(a) G is LL(k) but not LL(k+1) (b) G is LL(k+1) but not LL(k)
(c) G is both LL(k) and LL(k+1) (d) G is ambiguous.

(x) A production rule is of the form A→ BXY.
(a) Assume X→ bZ, b is a terminal. Then Follow(B) does not have b in it.
(b) Assume Y→ ε. Then Follow(X) contains Follow(A).
(c) If B→ ε, then Follow(A) contains First(X).
(d) If XY => ε, then Follow(A) contains Follow(B).

Group – B

2. (a) Draw FA for recognizing keyword IF in a source code written in C language.

(b) Write regular definition for the following:
i. All strings of lowercase letters that contain the five vowels in order.
ii. Comments, consisting of a string surrounded by /* and */. There will not be any intervening */. If it is inside double-quotes (" "), it will not be treated as a comment.

(c) A lexical analyzer uses the following patterns to recognize three tokens T_1, T_2, and T_3 over the alphabet {a, b, c}.
T_1: a?(b|c)*a T_2: b?(a|c)*b T_3: c?(b|a)*c
Note that 'x?' means 0 or 1 occurrence of the symbol x. Note also that the analyzer outputs the token that matches the longest possible prefix.

(c) Consider the following code which computes the inner product of two vectors:

```
prod := 0;
i := 1;
repeat {
    prod := prod + a[i] * b[i]
    i = i+ 1;
 until i > 20
}
```

i. Write the three address codes for the above code.
ii. Create basic blocks and the control flow graph for that IR.

(2 + 2) +(2 + 2) + 4 = 12

Group - E

8. (a) Explain the terms with example:
(i) Register descriptor (ii) Address descriptor.

(b) Translate the following code into machine code and show the register and address descriptors while the instructions are generated. Assume that two registers are available.
(i) x = y * z, (ii) w = p + y, (iii) y = y * z, (iv) p = w – x.

(2 + 2) + 8 = 12

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

(b)

```
int main()
{
    int n,k=0;
    scanf("%d",&n);
    for(i=2;i<n;i++)    {
       if(n%i==0)  break;
    }
    k=1;
    if(i==n)
        x=1;
    else
        x=0;
}
```

(i) Identify the basic blocks and the control flow graph.
(ii) Using dead code elimination, identify the statements that are eliminated.

(2 + 2 + 2) + (4 + 2) = 12

If the string *bbaacabc* is processed by the analyzer, which sequence of tokens does it output?

(d) Construct DFA directly from [not by generating NFA] the regular expression L = (a | b)* ab.

2 + (1.5 + 1.5) + 2 + 5 = 12

3. (a) Show the output of each phase of compiler on the following input.

```
int GCD(int a, int b) {
    while(a!=b) {
        if(a>b)  a=b;
        else b=a;      }
    return a;
}
```

(b) List the various error recovery strategies for lexical analysis.

9 + 3 = 12

Group – C

4. (a) Construct context free grammar for Roman numerals from 1-20. Draw a parse tree for xiv.

(b) Consider the following CFG, which has the set of terminals T = {stmt,{,},;}. This grammar describes the organization of statements in blocks for a fictitious programming language. Blocks can have zero or more statements as well as other nested blocks, separated by semicolons, where the last semicolon is optional. (P is the start symbol here)

P → S S → stmt | {B B → } | S} | S;B

i. Construct a DFA for viable prefixes of this grammar using LR(0) items.
ii. Identify any shift-reduce and reduce-reduce conflicts in this grammar under the SLR(1) rules.
iii. Assuming that an SLR(1) parser resolves shift-reduce conflicts by choosing to shift, show the operations of such a parser on the input string {stmt;}.

(2 + 1) + (3 + 3 + 3) = 12

5. (a) Explain the hash table organization of symbol table.

(b) What is phrase level error recovery in parsing?

(c) Differentiate static and dynamic scope of variables with example.

(d) E→TE' E' → +E | ε T→FT' T'→T | ε
F→ PF' F' → *F' | ε P→(E) | a | b | ε
(i) Construct the FIRST and FOLLOW set for all non-terminals.
(ii) Construct the predictive parsing table for the above grammar.

3 + 2 + 3 + (2 + 2) = 12

Group - D

6. (a) The following context-free grammar describes part of the syntax of a simple programming language. Non-terminals are given in capitals and terminals in lower case.
var represents a variable name and *const* represents a constant. The productions for ASSN-STMT are not given, as they do not affect the question.

PROGRAM → procedure STMT-LIST
STMT-LIST → STMT STMT-LIST | STMT
STMT → do var = const to const { STMT-LIST }| ASSN-STMT

i. Show the parse tree for the following:

```
Procedure
do i = 1 to 100 {
ASSN-STMT
ASSN-STMT
}
ASSN-STMT
```

ii. Create attribute(s) and add semantic functions to the above grammar that compute the number of executed statements for a program conforming to this grammar. Write them beside the productions above.
iii. Using the tree from part (i), compute the value of your attributes.

(b) Give a comparative discussion about quadruples, triples, and indirect triples representation for the following example. x = y * −z + y * −z

(c) Consider the following SDT. If an LR parser caries out the translations on an input string "*aabbccdb*", what is the output?

```
S→aaS  {print("a");}
   | bA  {print("b");}
   | b   {print("c");}
A→bcA  {print("d");}
   | cdS   {print("e");}
```

(1 + 3 + 1) + 4 + 3 = 12

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

(b) This grammar generates binary numbers with an optional negative sign (−) and optional decimal point:

S → N | − N N → L . L | L L → L B | B B → 0 | 1.

i. Design an L-attributed definition to compute S.val, the decimal number value of the input string. For example, the translation of 101.101 should be the decimal number 5.625.
ii. Draw parse tree for string −101.101 and annotate the nodes with values computed by the rules.