USN	1	M	S							

RAMAIAH
Institute of Technology

(Autonomous Institute, Affiliated to VTU)
(Approved by AICTE, New Delhi & Govt. of Karnataka)
Accredited by NBA & NAAC with 'A+' Grade

MAKEUP / SUPPLEMENTARY SEMESTER EXAMINATIONS SEPTEMBER 2022

Program	:	**B.E. : Computer Science and Engineering**	**Semester**	:	**VI**
Course Name	:	**Compiler Design**	**Max. Marks**	:	**100**
Course Code	:	**CS61**	**Duration**	:	**3 Hrs**

Instructions to the Candidates:

- Answer one full question from each unit.

UNIT- I

1.	a)	Explain the phases of the compiler and show the translation of each phase for the assignment statement. Assume all variables are integers. Eqn=(a-b)*(a-b)	CO1	(08)
	b)	Consider the alphabet Σ= {a,b}. Define a short, regular expression that generates strings over Σ that **contain exactly one "a" and atleast one "b".** Show the traces for the tokenization of input: **int a, b;** using the input buffering technique "sentinels".	CO1	(06)
	c)	Describe why elimination of Left recursion is required for Top Down Parsing. Eliminate the left recursion in the following grammar. **S→ A | B** **A→Aa | ε** **B→Bb| Sc | ε**	CO1	(06)
2.	a)	Differentiate the terms Token, Pattern and Lexeme. Write a pattern description for representing an identifier and give examples for token and its corresponding lexeme.	CO1	(07)
	b)	Construct transition diagram for recognizing arithmetic operators. Sketch the implementation for the transition diagram for the same.	CO1	(07)
	c)	Appraise with an example which data structure in a compiler is used for managing information about identifiers and their attributes?	CO1	(06)

UNIT – II

3.	a)	Illustrate the Top-down parsing of the input from left to right with an example.	CO2	(08)
	b)	Write the method and steps to construct the canonical LR(0) collection for a grammar using augmented grammar, CLOSURE() and GOTO() functions.	CO2	(07)
	c)	Discover the way of making shift-reduce decisions using LR (0) automaton.	CO2	(05)
4.	a)	Summarize the Recursive-descent top-down parsing method for syntax analysis with a set of recursive procedures to process the input.	CO2	(08)
	b)	Demonstrate the Behavior of the LR Parser using LR parse table with suitable examples.	CO2	(07)
	c)	Give the rules for constructing LR(1) ACTION and GOTO functions from the sets of LR(1) items.	CO2	(05)

UNIT – III

5. a) Illustrate how the parser stack is implemented for bottom up strategy with the help of the given desk calculator grammar — CO3 (08)

E→E+T | T
T→T*F | F
F→num

b) Consider the following attribute grammar — CO3 (06)

Production	Semantic Rule
$S \rightarrow L_1.L_2$	$S.v = L_1.v + \frac{L_2.v}{2^{L2.c}}$
$S \rightarrow L$	S.v=L.v
$L \rightarrow L_1\ B$	$L.v=L_1.v*2+B.v$ $L.c=L_1.c+B.c$
$L \rightarrow B$	L.v=B.v L.c=B.c
$B \rightarrow 0$	B.v=0 B.c=1
$B \rightarrow 1$	B.v=1 B.c=1

i. Draw the Annotated parse tree for the input '110.01', and show the dependency graph for the associated attributes.
ii. Describe one correct order for the evaluation of the attributes. Assume 'c' and 'v' are the synthesized attributes.
iii. What will be the value of S.v when evaluation has terminated?

c) Differentiate S-Attributed Definition and L-Attributed Definition — CO3 (06)

i. A->Q R { R.i=f(A.i); Q.i=f(R.i); A.s=f(Q.s); }
ii. A→ S R {R.i=S.s; A.s=S.s;}

Is the above definitions S-Attributed or L-attributed? Justify.

6. a) Write a S-attributed Definition for the given CFG to count the number of 1's in the given binary value. Draw an annotated parse tree for the input string **1110** — CO3 (07)

G: **N→D**
$D \rightarrow D_1$ B
D→B
B→0
B→1

b) What is an attribute called if it is associated with the nonterminal symbol on the left of a CFG rule, and its value is calculated using the attributes of the symbols on the right of the CFG rule? Explain with an example. — CO3 (05)

c) Explain the use of control stack in the activation of a procedure call. List and explain the contents of a general activation record. Construct the Activation Tree and Activation record for the recursive program given. Assume value of n=3. — CO3 (08)

```
main() {
        int num;
        num=sum(number);
}
int sum(int n) {
        if (n == 0)
                return 0;
        else
                return n + sum(n - 1);
 }
```

UNIT – IV

7.	a)	Describe about the directed acyclic graphs associations with syntax tree for an expression with examples.	CO4	(08)
	b)	Sub divide the ways of defining type expressions for declarations. Give examples.	CO4	(07)
	c)	Point out that the back patching can be used to generate code for Boolean expressions and flow of control statements in one pass.	CO4	(05)
8.	a)	Show that the three address code is a linearized representation of a syntax tree.	CO4	(08)
	b)	Discuss the generations of three address code for expressions incrementally.	CO4	(07)
	c)	Enumerate the translation of a switch statement using syntax directed translation.	CO4	(05)

UNIT – V

9. a) Write Three address code and construct the basic blocks for the following program segment — CO5 (08)

```
rev=0;
while(num>=0){
  dig= num%10;
  rev=rev*0 +dig;
}
```

b) Discuss in detail any three code improving transformations with example. — CO5 (06)

c) Compare the use of Address Descriptor and Register Descriptor. Convert the given three-address code into machine code assuming three registers are available. Show the register and address descriptors after each step. — CO5 (06)

t = x - y
w = x - z
v = t + w

10. a) Discuss about the primary tasks encountered in the design of a code generator. — CO5 (09)

b) Explain the implementation of getReg(I) function, for a three-address instruction, such as **I: a = b + c** — CO5 (05)

c) For the given block of code, perform code improving transformations assuming — CO5 (06)

i. if b is not live on exit from the block
ii. If both b and d are live on exit

Block of Code:
a = b + c
b = a - d
c = b + c
d = a - d

****