CS61

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

EXAMINATIONS SEPTEMBER /OCTOBER 2021
SUPPLEMENTARY SEMESTER / GRADE IMPROVEMENT/ RE –REGISTERED CANDIDATES

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 any five full questions.

1.	a)	Explain how the front end of a compiler works by translating the given statement Area=length*breadth;	CO1	(08)
	b)	Illustrate the processing of input buffering improvement technique "sentinels".	CO1	(07)
	c)	Explain how the tokens will be specified in the lexical analysis phase of the compiler.	CO1	(05)
2.	a)	Explain about the role of lexical analyzer. Why is the analysis phase of compiler separated into lexical analysis and parsing?	CO1	(08)
	b)	Eliminate Left Recursion from the following grammar: i. S→IeTS|IeTSeS|a T→ Tb|b ii. A→Sa S→Ab|Sc|b	CO1	(06)
	c)	Construct a transition diagram to recognize the following patterns. Bat, Cat, Mat, Hat, Have, His	CO1	(06)
3.	a)	Construct the predictive parsing table for the given grammar and show the parsing steps for the string "uvuvxz" S → uBz B → Bv | vuE | vxuE | ByE E → v | vx	CO2	(14)
	b)	What is meant by handle pruning? Show the working of a shift reduce parser for accepting $id_1 * id_2$, considering the grammar: E → E + T | T T → T * F | F F → id	CO2	(06)
4.	a)	Construct LR (1) parsing table for the grammar. S → L = R | R L → *R | id R → L	CO2	(12)
	b)	Explain the rules for computing FIRST(X) and FOLLOW(X). Illustrate using the grammar: E→TE' E'→+TE' |∈ T→FT' T'→*FT'|∈ F→(E) |id	CO2	(08)

5.	a)	Construct Activation Tree and Activation record for the recursive code fragment for finding factorial of a number.	CO3	(07)
	b)	Design an L-attributed SDD to compute D.val G: D→BX X→BX | ε Give an Annotated Parse Tree and Dependency Graph for the input 1101$.	CO3	(07)
	c)	Define postfix SDT's. Write the rules for turning an L-attributed SDD into an SDT.	CO3	(06)
6.	a)	Generate the SDT for the following grammar S→while (C) S;	CO3	(08)
	b)	Illustrate how infix to prefix translation is implemented with the help of semantic actions.	CO3	(06)
	c)	Explain inherited attributes and synthesized attributes with examples.	CO3	(06)
7.	a)	Obtain the DAG for the expression a+b+(a+b)+(a+b)*a. Also give the sequence of steps for constructing the same.	CO4	(08)
	b)	Describe the process of generating three address code for flow control statements with help of Annotated Parse tree for if(x<0 && x>4 || !x) x=1;	CO4	(07)
	c)	Write and explain the SDD for switch statements.	CO4	(05)
8.	a)	Translate the following three address codes into quadruple and triple. i. t1=a+b; t2=c/t1; t3=i*4; t4=a[t3]; s=t4 ii. t1=n*2; param t1; param 1; t2=call fun,2; t3=t2+2; f=t3	CO4	(06)
	b)	Write the algorithm for Unification of a pair of nodes in a type graph.	CO4	(06)
	c)	Design an SDD for computing the type and width of basic types and array types. Also Construct Annotated Parse tree for int [2][2].	CO4	(08)

9.	a)	What do mean by basic block write three address code , basic block and flow graph for the following c segment
		prod=0;
		i=1;
		do
		{
		prod=prod + a[i]* b[i];
		i=i+1;
		}
		while(i <=20);
			CO5	(10)

	b)	Describe the following terms with an example i) Common sub expression elimination ii) Dead code elimination	CO5	(06)
	c)	How can be the algebraic identities useful in code optimization.	CO5	(04)
10.	a)	Briefly explain the main issues in code generation.	CO5	(10)
	b)	Explain the code generation algorithm and generate code for the following expression. X=(a - b) + (a + c)	CO5	(10)

****