Course Code:	CST358	

Sixth Semester B.E (Computer Science and Engineering) Examination

COMPILER DESIGN

Time: 2 Hours]	[Max. Marks: 40

Instructions to Candidates:

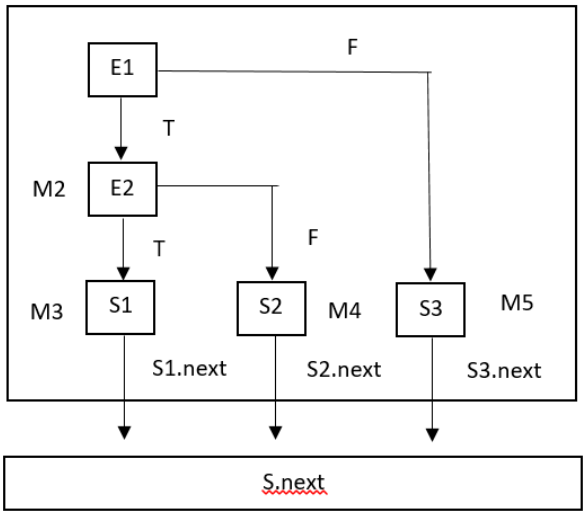
1. All questions carry marks as indicated against them.
2. Assume suitable data wherever necessary.

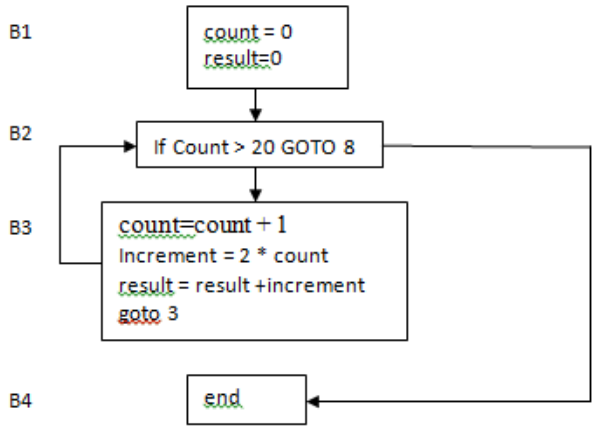
Question		Description of Question	Marks	CO
1	(a)	Construct the DFA corresponding to the regular expression using Direct method: (a|b)* ab*	(5M)	CO1
		--OR--		
	(b)	(i) Write a regular expression (RE) over the alphabet set {0,1} such that string contains zero or more 1s in the beginning followed by three 0s. (ii) Construct the DFA using direct method for the RE that acts as the lexical analyzer. (iii) Show how a valid input string can be accepted by this lexical analyzer.	(5M)	CO1
2	(a)	Construct CLR parsing table using LR(1) items, and Parse the string "aaba". S→ aA2 | aaA | Bb A→ Ba | a B→ b	(5M)	CO2
		--OR--		
	(b)	Construct LALR parsing table for the given grammar S→ CC C→ cC | d	(5M)	CO2
	(c)	Consider the given grammar and solve using LL(1) parser. Also state if the grammar is LL(1) or not. S→ SA | SB | a | bc A→ cA | Ca B→ b | ε C→ abB | abC | ε	(4M)	CO2

3	(a)	Construct the annotated parse tree and find the three-address code for the given construct.
		for(i=1;i<50; i+1)
		while(a>10) do
		a=a+1;
			(3M)	CO3

		--OR--		

	(b)	Construct the annotated parse tree and translate the following into three address code using SDTS for switch statement:
		switch(x)
		{
		case1: a=b+3;
		case2: switch(a+b)
		{
			(3M)	CO3

		case 7: b=a+2 } }		
	(c)	Consider the flow diagram specifying the flow for the given grammar. Write the semantic rules corresponding to the grammar in the form of SDTS, containing backpatch rules and S.Next. E denotes expression, S denote statements, M and N are used as address holders in the grammar, and T and F are True and False conditions for the expression. Grammar: S→ if E1 then M2 if E2 then M3 S1 N1 else M4 S2 N2 else M5 S3 Flow for S:  Assume: E1 as (a>b), E2 as (b>c), S1 as a=a+1, S2 as a=a+2 and S3 as a=a+3. Construct the annotated parse tree and find the intermediate Three Address Code (TAC).	(5M)	CO3
4	**(a)**	Construct the SLR parsing table using LR (0) items for the given ambiguous grammar. E→ E+E | E*E| id Find the type of parser conflicts and resolve the conflicts by giving precedence to * and using left association. Using the conflict free SLR table write the error routines to handle the errors and show how error is handled for the input string: **id + * id**	(5M)	CO2
		--OR--		
	(b)	Write the C code for merge sort. Draw the activation tree when numbers 5 8 1 9 4 2 7 3 are to be sorted. Also show the intermediate control stacks having the activation records.	(5M)	CO1

5	**(a)**	 Compute the dataflow equations for each block.	(3M)	CO4
		--OR--		
	(b)	Consider the given three address code. Generate the program flow graph and state the loops in the program flow. Compute the IN -OUT GEN KILL equation. 1. i=m-1 2. j=n 3. a=v1 4. i=i+1 5. j=j-1 6. if e1 goto (9) 7. i= v3 8. goto (10) 9. a=v2 10. if e2 goto (4)	(3M)	CO4

	(c)	Find the Program flow graph and find dominators in the following:
		a=0;
		b=1;
		c=2;
		L2: if (b>100) goto L3;
		a=a+1
		d=e+f
		L1: if (b>50) goto L3;
		c=a;
		g=10*d;
		h=g+c;
		b=b+2;
		goto L1;
		b=b+4;
		goto L2;
		L3: i=b;
			(2M)	CO4

6	**(a)**	Consider the expression: x=a/(b+c) – d*(e+f) Construct the DAG and find the minimum number of registers required. Also generate the target code using gencode() procedure of labeling algorithm.	(6M)	CO4