B.E. COMPUTER SCIENCE AND ENGINEERING
THIRD YEAR
SECOND SEMESTER EXAM 2019

Subject: **Compiler Design**

Time : Three hours **Full Marks: 100**

Group-1 (20 marks)	Answer any two questions: 1. (a) What are the different types of languages according to Chomskey's hierarchy? Briefly discuss all of them. Which of them are useful for compiler construction and why? 10 2. (a) Write a regular expression to represent a floating point constant with a mantissa and an exponent, for example 24.25e3. Construct an NFA for the regular expression using McNaughton-Yamada-Thompson algorithm. 10 3. Construct a DFA directly from a regular expression: a (a | b) * b b 10
Group-2 (40 marks)	Answer any two questions from this group. 4. (a) While constructing a parsing table, why do you need to construct the FIRST and FOLLOW sets? Explain your answer with examples. (b) Consider the grammar: $S \rightarrow a\ A\ B\ e$ $A \rightarrow A\ b\ c$ $A \rightarrow b$ $B \rightarrow d$ (a, b, c, d, e are terminals and S is the start symbol) (i) Construct a top-down parsing table for the grammar. (ii) Show the parsing actions for a string *"a b b c d e"*. (iii) Is the grammar LL(1)? 6+(8+4+2)=20 5. (a) With an example explain how do you remove indirect left recursion in a grammar. You may also use the algorithm to explain your answer, (however the algorithm is not necessary). (b) Consider the grammar: S --> ABC A --> $aA \mid \varepsilon$ B --> $bB \mid \varepsilon$ C --> c (Terminals = $\{a, b, c\}$, Non-terminals = $\{S, A, B\}$, Start Symbol = S) (i) Show the leftmost and rightmost derivations of two strings (at least four characters long) which are generated from the above grammar. (ii) Generate an LL(1) parsing table for the above grammar. Put error actions in the table. (c) When is left factoring important in top-down parsing? 6+(4+7)+3=20 6. (a) What are the *closure* and *goto* functions in LR parsing? (No definition is required, explain with an example). (b) Consider the following grammar: S --> AB S --> BA A --> aaB A --> a B --> bbA B --> a (Terminals = $\{a, b\}$, Non-terminals = $\{S, A, B\}$, Start Symbol = S) Construct LR(0) item set for the above grammar. (c) Construct the SLR parsing table for the above grammar.

	(d) Show the actions of the parser for the input string: (aabba). 4+6+6+4=20
Group-3 (30 marks)	Answer any two questions from this group. 7. (a) What is *syntax-directed definition (SDD)*? What are synthesized and inherited attributes? (b) Write an SDD for declaration of list of variables (consider the types 'integer', 'real' and 'char'. Using the SDD, give an annotated parse tree for a declaration statement of two variables of type 'real'. Also draw a dependency graph. (c) What is an L-attributed definition? 5+(3+2+2)+3=15 8. (a) How do you evaluate synthesized attributes using bottom-up parsers? (b) What are translation schemes? (c) Using examples explain how do you evaluate translation schemes for L-attributes definitions? (Use markers for this purpose). 4+3+8=15 9. (a) What is the use of symbol table? In which phases of compiler a symbol table is used? (b) What are the operations performed on a symbol table when it is implemented as a single hash table? (c) What is a three-address code? What is 'static single assignment'? How does it differ from three-address code? (d) Give an example of a 'control-flow graph'. 3+4+6+2=15

Group-4 (10 marks)	Answer any one question
	10. (a) What are the main tasks in 'code generation'?
	(b) With an example explain what are 'liveness' and 'next use' of variables. Why do you need this information?
	(c) Discuss the use of 'Register Descriptor' and 'Address Descriptor'. 2+5+3=10
	
	11. (a) Optimize the following code and discuss each optimization technique that you have applied stating their advantages:
	
	`for (i = 0, i<n; i++)`
	`{`
	`  for (j=0; j<n; j++)`
	`  {`
	`  if (i%2)`
	`  {`
	`  x += (4*j + 5*i);`
	`  y += (7 + 4*j);`
	`  }`
	`  }`
	`}`
	
	(b) With examples discuss the difference between 'useless code (dead code) elimination and unreachable code elimination.
	(c) What are the different scopes of code optimization?
	5+3+2=10