Roll No. □□□□□□□□□□

ANNA UNIVERSITY (UNIVERSITY DEPARTMENTS)

B.E. / B. Tech / B. Arch (Full Time) - END SEMESTER EXAMINATIONS, NOV/DEC 2024

INFORMATION TECHNOLOGY
Fifth Semester (V)
IT5502 & Compiler Engineering
(Regulation 2019)

Time: 3hrs Max.Marks: 100

CO 1	Understand the concept of lexical analysis and construction of deterministic and non-deterministic automata.
CO 2	Understand the concept of parsing and construction of parser.
CO 3	Develop an Intermediate Code generator.
CO 4	Study programming language design, target machine design and run time environment of compilers.
CO 5	Study about the compiler construction tools.
CO 6	Obtain knowledge to construct a prototype compiler for a subset of a programming language.

BL – Bloom's Taxonomy Levels
(L1 - Remembering, L2 - Understanding, L3 - Applying, L4 - Analysing, L5 - Evaluating, L6 - Creating)

PART- A (10 x 2 = 20 Marks)
(Answer all Questions)

Q. No	Questions	Marks	CO	BL
1	Identify the regular expression for the language over input alphabets $\Sigma = \{p,q\}$ consisting of strings where atleast 2 p's in the beginning , atleast one q in the last and alternate p and q in the middle	**2**	CO1	L3
2	Write character classes for the following: a) A string consisting of lower case vowels having 3 numbers in between second and third vowel. b) Digits in hexadecimal format.	2	CO1	L3

3	Identify the lexeme, pattern and tokens from the following source code:
	void result()
	{
	int a , b, c;
	sum = a * b + 5;
	}
		2	CO6	L3

4	List out the various lexical errors and recovery actions performed by the lexical analyzer.	2	CO6	L2
5	Give a leftmost and rightmost derivation for the string ((b,b),b,(b)) using the following grammar:	2	CO2	L3

	S-> (L) | b L-> L, S| S			
6	Perform Left Factoring for the following grammar: A->ad|a|ab|abc|b	2	CO2	L3
7	Translate the arithmetic expression c= p * -q + b[i] into Quadruples, Triples and Indirect Triples.	2	CO3	L3
8	Construct DAG for the following arithmetic expression: a*a + (b + a*a + (a + (a*b)))	2	CO3	L3
9	Using Peephole Optimization, how unreachable code and unnecessary jumps can be eliminated in intermediate representation? Illustrate with an example	2	CO5	L2
10	How do you bootstrap a compiler? Give an example.	2	CO5	L2

PART- B (5 x 13 = 65 Marks)

(Restrict to a maximum of 2 subdivisions)

<table>
<tr><th>Q. No</th><th>Questions</th><th>Marks</th><th>CO</th><th>BL</th></tr>
<tr><td>11 (a) (i)</td><td>Construct epsilon Non Deterministic Finite Automata for the following regular expression and convert it into DFA by applying subset construction method. Also apply State Minimization algorithm to reduce the total number of states in DFA.
(c | d)* dd(c|d)</td><td>9</td><td>CO1</td><td>L3</td></tr>
<tr><td>(ii)</td><td>A lexical analyzer needs to recognize the patterns for the strings such as “result” and “sum” (identifiers) and signed numbers such as -45.30 and -64.503E+2:
a) Give the regular definitions for the above mentioned strings.
b) Draw the state transition diagram for signed numbers and identifiers.</td><td>4</td><td>CO1</td><td>L3</td></tr>
<tr><td colspan="5">OR</td></tr>
<tr><td>11 (b) (i)</td><td>Construct DFA for the following regular expression using syntax tree approach.
(a| Ɛ)* ac*|bc*b</td><td>9</td><td>CO1</td><td>L3</td></tr>
<tr><td>(ii)</td><td>Consider the following Ɛ- transitions (Transition table) with respect to the states p,q,r and inputs Ɛ, a, b and c:
<table>
<tr><th>State/Input</th><th>Ɛ</th><th>a</th><th>b</th><th>c</th></tr>
<tr><td>State p</td><td>{q,r}</td><td>∅</td><td>{q}</td><td>{r}</td></tr>
<tr><td>State q</td><td>∅</td><td>{p}</td><td>{r}</td><td>{p,q}</td></tr>
<tr><td>State r</td><td>∅</td><td>∅</td><td>∅</td><td>{r}</td></tr>
</table>
a) Visualize NFA from the table.
b) Convert it into DFA and identify its regular expression.</td><td>4</td><td>CO1</td><td>L3</td></tr>
<tr><td>12 (a) (i)</td><td>Explain the phases of a Compiler in detail and write down the output of each phase of the compiler for the given expression result = a + b * c / 4.</td><td>9</td><td>CO6</td><td>L3</td></tr>
<tr><td>(ii)</td><td>Consider the language R = {ab, bba, aa, ab} and S = {0,11,10,1}. Compute the following
a) S|(RS)
b) R(RUS)*</td><td>4</td><td>CO1</td><td>L2</td></tr>
<tr><td colspan="5">OR</td></tr>
<tr><td>12 (b) (i)</td><td>Write a Lex program to find the total occurrence of the user id and password in the given source text. Write the identified user id and password in a separate file.</td><td>9</td><td>CO6</td><td>L3</td></tr>
</table>

	User id consists of 3 or more alphabets and 3 numbers or a mail id restricted to a length of 12 characters. Password consists of an initial capital letter followed by one or more small letters followed by a special character ($/&/#/@) restricted to a length of 10 characters.			
(ii)	Illustrate how a pair of input buffers can handle a large look ahead safely with a neat diagram. What is the role of sentinels in buffer pair? Explain.	4	CO1	L2
13 (a)	Construct a Predictive Parser and parsing table for the following grammar: S-> AB A-> x,A | (B) B-> y*B | z Check whether the given string ‘x,x,(y*z)z” belongs to the above grammar.	13	CO2	L3
	OR			
13 (b)	Construct a CLR parser for the following grammar: S -> BB* B-> cB B->d Parse the string “cccdd*” using the above grammar.	13	CO2	L3
14 (a) (i)	Simulate the steps of constructing the Syntax tree and DAG for the following simple expression using appropriate S-Attributed SDD: c = (a-4) + b * (d+e) * f	9	CO3	L3
(ii)	Write down the rules which are used by the compiler for Type Checking with a suitable exampl ɜ. Also list out the various Type conversions that takes place during compilation time using suitable algorithm.	4	CO3	L3
	OR			
14 (b) (i)	Write down the semantic actions for the Translation of Arithmetic Expressions and generate the three address code for the following expression using annotated parse tree: sum = y * z + (- b * c) / w	9	CO3	L3
(ii)	With the proper L-attributed definition, draw a dependency graph with evaluation order for the following statement: 8 + 4 – (3 *6)	4	CO3	L3
15 (a) (i)	Explain the issues involved in the design of a Code Generator in detail with suitable examples.	8	CO4	L2
(ii)	Generate intermediate code and target code for the following C snippet given below and estimate the cost of the final target code: { sum = a[i] + b[i]; d = sum + *p; } (Assume array entries require 8 bytes and p is a memory location.)	5	CO4	L5
	OR			

15 (b) (i)	Generate three address code and draw the flow graph with proper identification of basic blocks and loops for the following C code used for sorting:
	for (i =1; i<n; i++)
	{
	key = arr[i];
	j = I -1;;
		8	CO4	L3

	while (j>=0 && arr[j] > key)
	{
	arr[j+1] = arr[j];
	j = j -1;
	}
	arr[j+1] = key;
	}
	(Assume array entries are numbers which require 8 bytes.)			

(ii)	Optimize the three address code generated in 15.b.(i) globally as well as locally by preserving the semantics of the original program.	5	CO4	L5

PART- C (1 x 15 = 15 Marks)

(Q.No.16 is compulsory)

Q. No	Questions	Marks	CO	BL
16. (i)	With appropriate SDD for flow control statements and Boolean expressions, translate the following conditional statement into three address code: if ((a>b && a>c) || (e<f || c !=d) , x =100;	10	CO3	L4
(ii)	Generate code for the same statement in one pass using Backpatching SDD.	5	CO3	L4