	ITER, SIKSHA 'O' ANUSANDHAN (Deemed to be University)		Assignment
Branch	**CSE, CSIT**	Programme	**B.Tech**
Course Name	**Compilers : Principles, Techniques and Tools**	Semester	**VI**
Course Code	**CSE 3739**	Academic Year	**2024-25**

Theory Assignment - 1

Date of Submission: On or before **09-04-2025**

Learning Level (LL)			
	L1: Remembering	**L3:** Applying	**L5**: Evaluating
	L2: Understanding	**L4**: Analysing	**L6**: Creating

Q's	Questions	COs	LL
1	Show the different inputs and outputs of the various phases of a compiler for the assignment statement **C= (F - 32) * (5/9)**	CO1	L1, L2
2	Provide regular expressions to characterize the following lexical items: a. Identifiers: alphanumeric strings starting with an alphabet, and can also contain special character '_' where you allow it is up to you, but state your decisions in your documentation. **Examples** *x, ab_2y, c2* etc. b. Numbers, which include integers, fixed point and floating point numbers. Leading zeros and redundant trailing zeros are disallowed; e.g., 002 is invalid, 2.00 is invalid but 2.0 is valid. Real numbers can be represented in the E (exponential) format as well (e.g. 2.0E-2 is valid and represents 0.002). Decisions whether 2.0E+2 is valid are up to you. c. white space (sequences of blank, tab, newline) d. Arithmetic operators: **+, -, /, *, div, mod** e. Logical constants: **true**, **false** f. Logical operators: **not**, **and**, **or** g. Comparison operators: **<=, >=, <, > , =** h. Parentheses: **(,)** i. Keywords: **if, then, else** j. Literal: A literal is anything that starts with " symbol and ends with " symbol (e.g. "core dumped")	CO2	L2, L3
3	A grammar for branching statements is defined as: *stmt* → ***if*** *expr* ***then*** *stmt* | ***if*** *expr* ***then*** *stmt* ***else*** *stmt* | ε *expr*→ *term* ***relop*** *term* | *term* *term* → ***id*** | ***number*** The pattern for the token relop is defined as: ***relop*** → < | > | < = | >= | = | < > Construct the state transition diagram that will recognize the token ***relop.***	CO2	L1, L3
4	Design the DFA for the given Regular Expression **a (a | b)* a** a) Convert the RE to Ɛ-NFA b) Convert Ɛ-NFA to NFA and NFA to Minimized DFA	CO2	L2, L3
5	Divide the following C program into appropriate lexemes and count the number of tokens. ***float limitedSquare (x)*** ***{*** ***float x;*** ***/* returns x-squared, but never more than 100 */*** ***return (x < = -10.0 || x>= 10.0)? 100 : x*x;*** ***}***	CO2	L2, L3

6	Consider the context free grammar: $S \rightarrow (L) \mid a$ $L \rightarrow L, S \mid S$ and the string $((a, a), a, (a))$ a) Give the leftmost derivation for the string. b) Give the rightmost derivation for the string. c) Give a parse tree for the string. d) Is the grammar ambiguous or unambiguous? Justify your answer. e) Eliminate the left recursion from the grammar (if any)	**CO3**	**L1, L3**
7	Let G be the following context-free grammar, where E is the start symbol of G ; **E→ E > E | E ≥ E | E < E | E ≤ E | E << E |E >> E | E != E** **| id** Construct an unambiguous grammar G' for above ambiguous G with reference to the operator table given below. **operator** | **Associativity** | **precedence** >, ≥, <, ≤ | left to right | lowest <<, >> | right to left | ↓ != | left to right | highest	**CO3**	**L2, L3**
8	Let G be the following Context-free Grammar, **P→ xQRS** **Q→ *yz* |z** **R→ w | Ɛ** **S→ *y*** Compute the FIRST and FOLLOW function for all the non-terminals present in the above Grammar.	**CO3**	**L2, L3**
9	Consider the following grammar. $N \rightarrow XY \mid YX$ $X \rightarrow a \mid ZXZ$ $Y \rightarrow b \mid ZY$ $Z \rightarrow a \mid b$ a) Construct the LL (1) parsing table. b) Verify the given grammar is LL (1) or not. c) Verify the given grammar is ambiguous or not.	**CO3**	**L3, L4**
10	Consider the following grammar: $S \rightarrow +SS \mid *SS \mid a$ and the string $+a*aa$ a) Construct the SLR sets of items of the above (augmented) grammar. b) Compute the GOTO function for these sets of items. Show the parsing table for this grammar. Is the grammar SLR?	**CO3**	**L3, L4**

Course Outcomes	CO1	Understand the overview of programming languages, language processors and the structure of a compiler.
	CO2	Acquire knowledge in theory of computation and their role in designing different types of tokens generated by lexical analyzer.
	CO3	Understand the role of Parser(s) (LL, SLR, CLR and LALR) and its types i.e. Top-down and Bottom-up parsers.
	CO4	Apply and evaluate syntax directed translation schemes, synthesized attributes, inherited attributes, and different techniques for symbol table organization
	CO5	Analyze the generation of various intermediate codes and the process of their optimization.
	CO6	Understand the target machine's run time environment, and its instruction set for code generation and techniques used for code optimization