Satyug Darshan Institute of Engineering and Technology, Faridabad

	Programme: B.TECH	**Year/Semester : 3rd/5th**
	Subject Name : Formal Languages, Automata and Compiler Design	**Subject Code:PCC-CS-502**

Prepared By: Ms. Vaishali

Unit -1

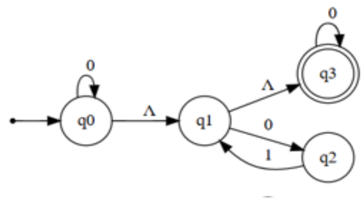
Short Questions

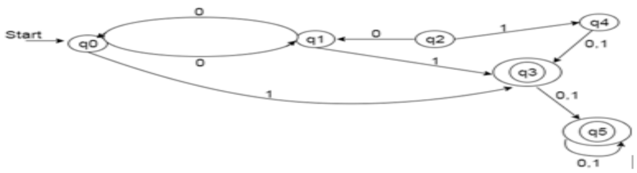
S. No.	Questions	Bloom Taxonomy
1	Define the terms: alphabet, string, and language with supporting examples.	K1
2	Define Kleene closure and positive closure.	K1
3	Define grammar and identify its four fundamental components.	K1
4	Define the term "sentential form."	K1
5	Define production rules within the context of formal grammars.	K1
6	Define a Context-Free Grammar (CFG).	K1
7	Define Context-Free Language (CFL).	K1
8	Define a Pushdown Automaton (PDA).	K1
9	Define a Context-Sensitive Language.	K1
10	Define a Turing Machine and its mathematical components.	K1
11	Summarize the meaning of a derivation in formal language theory.	K2
12	Interpret the concept of Epsilon-transition in automata.	K2
13	Summarize the definition of a regular expression.	K2
14	State the criteria for language acceptance in finite automata.	K2
15	Summarize the definition of a Linear Bounded Automaton (LBA).	K2
16	Define the characteristics of a recursively enumerable language.	K2

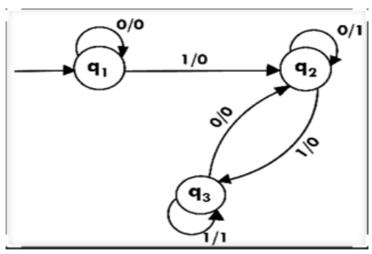
Long Questions

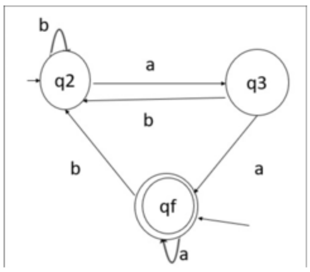
S. No.	Questions	Bloom Taxonomy
1	Construct a grammar capable of generating the language **L = {a^n b^n | n >= 1\}**	K3
2	Design a Deterministic Finite Automaton (DFA) to accept strings ending with the suffix "01".	K3
3	Construct a Context-Free Grammar (CFG) for the language L={ww^R | w belongs to {0,1\}*}.	K3
4	Design a Pushdown Automaton (PDA) to recognize the language a^nb^n.	K3
5	Design a Turing Machine to accept strings of the form 0^n1^n.	K3
6	Demonstrate the step-by-step working of a Turing Machine for palindrome recognition.	K3
7	Execute the conversion of a Context-Free Grammar into Chomsky Normal Form (CNF).	K3
8	Distinguish between leftmost and rightmost derivations using a specific example.	K2
9	Explain the hierarchical relationship between alphabets, languages, and grammars.	K4
10	Compare different types of derivations through illustrative examples.	K4
11	Relate the structure of parse trees to the process of derivation.	K4
12	Prove that every language generated by a grammar belongs to the class of recursively enumerable languages.	K4
13	Categorize the four types of grammars according to the Chomsky Hierarchy.	K4
14	Differentiate between DFA and NFA in terms of structure and transition functions.	K4
15	Illustrate the Chomsky hierarchy using specific language examples for each level.	K4
16	Examine the concept of ambiguity in CFG and provide supporting examples.	K4
17	Contrast Chomsky Normal Form (CNF) and Greibach Normal Form (GNF).	K4
18	Distinguish between deterministic and nondeterministic Pushdown Automata.	K4
19	Compare the capabilities of finite automata, PDA, and Turing Machines.	K4

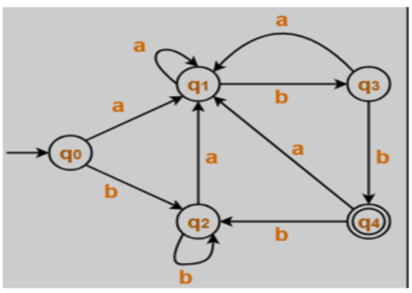
<table>
<tr><td>20</td><td>Explain the concept of decidability and its implications in computation.</td><td>K4</td></tr>
<tr><td>21</td><td>Construct a DFA over the alphabet Σ={0,1} that accepts only those strings where the number of 0s is a multiple of 3.</td><td>K3</td></tr>
<tr><td>22</td><td>Transform the following grammar into Chomsky Normal Form: S→ASB|ϵ,A→aAS|a,B→SbS|A|bb.</td><td>K3</td></tr>
<tr><td>23</td><td>Design a PDA that accepts the language L={wcwR|w∈{a,b}∗} by empty stack; provide the full transition table.</td><td>K3</td></tr>
<tr><td>24</td><td>Apply the Pumping Lemma for Regular Languages to prove that L={a^nb^n|n≥0} is not a regular language.</td><td>K4</td></tr>
<tr><td>25</td><td>Construct a transition diagram for a Turing Machine that accepts the language L={0^n1^n ∣ n≥1} and simulate the input 0011.</td><td>K4</td></tr>
<tr><td>26</td><td>Convert the following NFA with null moves in NFA without null moves-

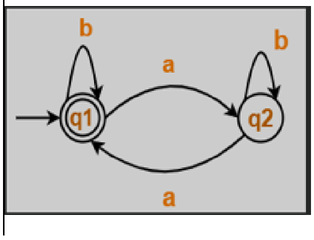
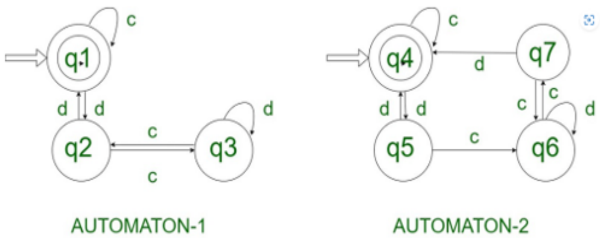

</td><td>K2</td></tr>
<tr><td>27</td><td>Construct FA by minimizing the given Finite Automata.


</td><td>K3</td></tr>
<tr><td>28</td><td>Solve the given machine to find its equivalent machine.


</td><td>K3</td></tr>
<tr><td>29</td><td>Construct RE using Arden's Theorem.


</td><td>K3</td></tr>
<tr><td>30</td><td>Minimize the given FA using any method


</td><td>K2</td></tr>
</table>

31	Find Regular Expression of the following DFA 	K2
32	Check whether the given FA are equivalent or not. 	K2
33	Using PDA, solve Language L=WWR, where **W€{0,1}***	K3

Unit -2

S. No.	Short Questions	Bloom Taxonomy
1	List the sequential phases of a compiler.	K1
2	Define the specific terms: token, lexeme, and pattern.	K1
3	Define the term "scanner generator."	K1
4	Identify Lex and Flex in the context of compiler construction.	K1
5	Define the concept of a parse tree.	K1
6	Define the properties of an LL(1) grammar.	K1
7	Define LR parsing.	K1
8	Define the scope of semantic analysis.	K1
9	Define synthesized and inherited attributes.	K1
10	Define a syntax-directed definition (SDD).	K1
11	Summarize the primary function of lexical analysis.	K2
12	State the purpose of syntax analysis in the compilation process.	K2
13	Summarize the meaning of ambiguity within a grammar.	K2
14	Interpret the concept of operator precedence parsing.	K2
15	Summarize the definition and utility of an attribute grammar.	K2

Long Questions

1	Write a Lex program to recognize and distinguish between identifiers and keywords.	K3
2	Demonstrate the process of LL(1) parsing using a specific grammar example.	K3
3	Illustrate all phases of a compiler using a comprehensive block diagram.	K3
4	Describe the difference between a Shift-Reduce conflict and a Reduce-Reduce conflict in an LR parser.	K2
5	Given the grammar $E \rightarrow E+T \mid T$, $T \rightarrow id$, perform Left Recursion elimination and calculate the FIRST and FOLLOW sets.	K3
6	Construct the Canonical Collection of LR(0) items for the grammar $S \rightarrow (S) \mid a$ and determine if the grammar is LR(0).	K4
7	Given a table of precedence relations (<,>,=), parse the input string id+id∗id and show the status of the stack at each step.	K4
8	Calculate the FIRST and FOLLOW sets for all non-terminals in the grammar: $S \rightarrow AaAb \mid BbBa$, $A \rightarrow \epsilon$, $B \rightarrow \epsilon$.	K3
9	Construct the LL(1) Parsing Table for the grammar: $E \rightarrow TE'$, $E' \rightarrow +TE' \mid \epsilon$, $T \rightarrow FT'$, $T' \rightarrow *FT' \mid \epsilon$, $F \rightarrow (E) \mid id$.	K3
10	Construct the SLR(1) parsing table for the grammar $S \rightarrow L=R \mid R$, $L \rightarrow *R \mid id$, $R \rightarrow L$. Identify any Shift-Reduce conflicts.	K3
11	Compute the closure of the item $S \rightarrow \cdot AA$ and generate the complete set of Canonical LR(0) items for the grammar $S \rightarrow AA, A \rightarrow aA \mid b$.	K4
12	Compute the set of LR(1) items for the following grammar and show the lookahead for each item: $S \rightarrow CC, C \rightarrow cC \mid d$.	K3

13	Construct the SLR(1) parsing table for the grammar S→AS∣b,A→a. Determine if any cells contain multiple entries.	K3
14	Analyze the grammar S→L=R∣R,L→∗R∣id,R→L. Prove that it contains a Shift-Reduce conflict in SLR(1) but is conflict-free in CLR(1).	K4
15	Using the CLR(1) items for the grammar S→AA,A→aA∣b, demonstrate the process of merging states to form the LALR(1) parsing table.	K4
16	Categorize the following grammars as SLR(1), LALR(1), or CLR(1) based on their ability.	K4
17	Simplify Grammar using LL(1) Parser. **E → TE', E'→ +TE' | ϵ, T → FT', T' → *FT' | ϵ, F → id | (E)**	K3
18	Can we simplify the given grammar using SLR(1) Parser **S → AaAb | BbBa, A→ ϵ, B→ ϵ** If not, solve it using appropriate parser.	K2
Unit -3		
S. No.	**Short Questions**	**Bloom Taxonomy**
1	Define a symbol table and its purpose	K1
2	Define the term "intermediate code" and its position in the compilation process.	K1
3	List the common types of intermediate representations used by compilers.	K1
4	Define a "basic block" in the context of code analysis.	K1
5	Define pipeline optimization and cache optimization.	K1
6	Summarize the primary objectives of code optimization.	K2
7	State the goals of loop optimization in improving program efficiency.	K2
8	Summarize the definition of target code generation.	K2
9	Identify the language generated by the grammar: S→aS∣bS∣ϵ.	K2
10	Define the structure of a "Triple" versus a "Quadruple" in three-address code representation.	K1
11	Identify the difference between "Local Optimization" and "Global Optimization" in the context of basic blocks.	K1
12	Contrast "Loop Unrolling" with "Loop Jamming" (Fusion) in terms of their impact on cache performance.	K2
Long Questions		
1	Demonstrate both leftmost and rightmost derivations for the string aabb using the grammar S→aSb∣ab.	K3
2	Illustrate the target code generation process through a specific code example.	K3
3	Show the step-by-step application of loop optimization for cache memory performance.	K3
4	Explain various intermediate code forms, including Three-Address Code and Syntax Trees.	K2
5	Translate the expression a = b + c * d / (e - f) into Three-Address Code (TAC) and then represent it using a Directed Acyclic Graph (DAG).	K3
6	Write short notes on a. Peephole Optimization b. Intermediate Code Generation	K2