**Assignment 2**

**Assignment Guidelines:**

1. Deadlines should be kept in mind. No extension in assignment dates would be given. No late submission will be accepted.
2. This is a group assignment. **PLAGIARISM IS NOT ACCEPTABLE AT ALL!** Zero marks will be given in case of plagiarism.
3. Deadline =28-02-25, Deadline is hard and firm.

|  |
| --- |
| **Question no 1:** |
| **Extracting Lexemes from Code and Tokens**  (a) Define what a lexeme is in compiler construction and explain its role in lexical analysis. (b) Consider the following code snippet and extract its lexemes along with their corresponding token types: int main() { int x = 10; float y = 5.5; return x + y; }  . |

|  |
| --- |
| **Question no 2:** |
| **Symbol Table Creation of Statements**   1. Explain the purpose of a symbol table in a compiler and its significance during lexical analysis. 2. Create a symbol table for the following statements: i) char c = 'x'; double d = 12.34; ii) int courses; char course1[ ]=”DataBase”   char course2[ ]=”Operating System” |

|  |
| --- |
| **Question no 3:** |
| **Finite Automaton (FA) of Words**   1. Construct a finite automaton (FA) for recognizing identifiers (variable names) in a programming language. 2. Draw a transition diagram for an FA that recognizes the words "lexicalanalysis" and "lexprogram". |

|  |
| --- |
| **Question no 4:** |
| **Table Encoding of FA**   1. Design a finite automaton (FA) for vending machine and encode its table. 2. Construct a transition table for the FA created in **Q4(a)**. |
| **Question no 5:** |
| **Minimized DFA Transition Diagram**  (a) Describe the process of minimizing a Deterministic Finite Automaton (DFA). (b) Minimize the DFA for the following regular expressions and provide the transition diagrams:   * a(b|c)\*d * (01|10)\* |

|  |
| --- |
| **Question no 6:** |
| **Diagram for Keywords and Symbols**     1. Draw a transition diagram for recognizing keywords such as if, else, while, and return.      1. Construct a transition diagram to recognize common symbols in a programming language   (e.g., +, -, \*, /, =, ;). |

|  |
| --- |
| **Question no 7:** |
| **Lex Programs**  i)Write a Lex program to identify keywords, identifiers, and numerical constants in your code.  ii)Write a Lex program to identify arithmetic operators (+, -, \*, /). |

|  |
| --- |
| **Question no 8:** |
| Write a **recursive descent parser** for the following grammar and demonstrate its working by parsing the input string:    w=id+id∗idw = id + id \* idw=id+id∗id  **Grammar:**   1. S → A 2. A → A+B ∣ B 3. B → B∗C ∣ C 4. C → id |

|  |
| --- |
| **Question no 9:** |
| Convert the following **context-free grammar (CFG)** into its **LL(1) form**, and then write a recursive descent parser for it.  **Grammar:**  S → Sa ∣ aS ∣ b |

|  |
| --- |
| **Question no 10:** |
| Construct the **LL(1) Parsing Table** for the following grammar and check if it satisfies the **LL(1) condition**.  **Grammar:**   1. S → AB 2. A → aA ∣ ϵ 3. B → bB ∣ ϵ |