**A blue and white logo

Description automatically generated**

**Name: Mubashir Ali**

**Reg # FA21-BCS-009**

**Course Name: Compiler Construction**

**Submitted To: Syed Bilal Haider Bukhari**

**Date: 03 Jan 2025**

**Lab Terminal**

**Question 2:** Explain two core functions of my Mini C Compiler.  
  
**Answer:** Based on the typical architecture of a Mini C Compiler, two core functions are:

1. **Lexical Analysis (Tokenization):**
   * **Purpose:** Converts the raw source code into a sequence of tokens, which are the basic building blocks like keywords, identifiers, operators, and literals.
   * **Process:**
     + Reads the source code character by character.
     + Groups characters into meaningful sequences called lexemes.
     + Classifies lexemes into token types (e.g., identifier, number, operator).
   * **Example:**
     + Source Code: int sum = a + b;
     + Tokens: int (keyword), sum (identifier), = (operator), a (identifier), + (operator), b (identifier), ; (delimiter).
2. **Syntax Analysis (Parsing):**
   * **Purpose:** Analyzes the sequence of tokens to ensure they follow the grammatical structure of the C language, constructing a parse tree or abstract syntax tree (AST).
   * **Process:**
     + Takes the sequence of tokens from the lexical analyzer.
     + Applies grammatical rules to check for correct syntax.
     + Builds a hierarchical tree structure representing the nested syntactic structure of the source code.
   * **Example:**
     + Tokens: int sum = a + b ;
     + Parse Tree: Represents that sum is an integer variable being assigned the result of adding a and b.

These functions are fundamental in transforming human-readable C code into a form that a machine can understand and execute.