**Q1: Explain code functions of your mini compiler**

**Functionalities of the Mini-C Compiler**

1. **Lexical Analysis (Tokenizer):**
   * Breaks the source code into tokens such as keywords, identifiers, operators, literals, and delimiters.
   * Identifies and categorizes lexical units in the input program.
2. **Syntax Analysis (Parser):**
   * Validates the grammatical structure of the tokenized code using predefined grammar rules.
   * Constructs a parse tree or an abstract syntax tree (AST) to represent the program’s syntax.
3. **Semantic Analysis:**
   * Ensures that the program’s statements are semantically correct.
   * Examples:
     + Type checking (e.g., verifying variable types in expressions).
     + Scope validation (e.g., variable declarations and usage).
4. **Intermediate Code Generation:**
   * Converts the source code into an intermediate representation (IR), such as three-address code (TAC) or quadruples.
   * Simplifies optimization and target code generation.
5. **Code Optimization:**
   * Performs optimizations to make the generated code more efficient.
   * Examples:
     + Constant folding (precomputing constant expressions).
     + Dead code elimination (removing unreachable or unused code).
6. **Code Generation:**
   * Translates the intermediate code into target machine code or assembly language.
   * Generates executable code that can run on a specific platform.
7. **Symbol Table Management:**
   * Maintains information about variables, functions, and other symbols.
   * Tracks attributes such as scope, type, and memory location.
8. **Error Handling:**
   * Provides detailed error messages for lexical, syntactic, and semantic errors.
   * Includes line numbers and descriptions of the issues.
9. **Input and Output Handling:**
   * Reads source code from an input file.
   * Writes the generated intermediate code, assembly code, or output to files.
10. **Execution (Optional):**
    * Includes a runtime environment or interpreter for executing the generated code.