Group Members

Aiman Khatoon (Fa20-bcs-017)

Mahnoor (Fa20-bcs-045)

Submitted To: Sir Bilal Haider Bukhari

Date of Submission: 28-Dec-2023

Lab Terminal

Question No4:

how functions works. Step by step:

Lexical Analysis:

Input: Source code in the form of a character stream.

Functionality: a. Tokenization: A function reads the input character stream and breaks it into tokens based on predefined lexical rules using regular expressions and finite automata. b. Categorization: Another function categorizes each token into types such as keywords, identifiers, literals, and operators. c. Token Stream Generation: A function generates a stream of tokens representing the input source code, and this token stream is passed to the next stage of the compiler.

Semantic Analysis:

Input: Token stream from the lexical analysis stage.

Functionality: a. Symbol Table Construction: A function constructs a symbol table to store information about identifiers, including their types and scopes. b. Type Checking: Functions are implemented to check the compatibility of operands and expressions, ensuring that the program adheres to semantic rules. c. Error Detection and Reporting: Functions detect and report semantic errors, such as undeclared variables or type mismatches, providing meaningful error messages to aid developers. d. Scoping and Lifetime Management: Functions manage scoping rules and handle variable lifetimes, ensuring correct variable access and lifetime management.

Syntax Analysis (if applicable):

Input: Token stream or parse tree from previous stages.

Functionality: a. Parse Tree Generation: If not covered by lexical and semantic analysis, a function generates a parse tree based on context-free grammars and parsing techniques such as LL or LR parsing. b. Syntactic Correctness Verification: Functions verify the syntactic correctness of the source code by analyzing the structure of the parse tree or token stream.

Error Handling:

Input: Detected errors during lexical, semantic, or syntax analysis.

Functionality: a. Error Reporting: Functions report errors with meaningful messages, aiding developers in identifying and fixing issues. b. Error Recovery Strategies: Functions implement strategies to gracefully recover from errors and continue the compilation process.

Documentation and Testing:

Input: Compiler components, algorithms, and test cases.

Functionality: a. Documentation Functions: Functions generate comprehensive documentation detailing design decisions, algorithms, and user guidelines. b. Testing Functions: Functions execute a suite of test cases to validate the correctness and performance of the compiler.

User Interface (optional):

Input: User commands or source code files.

Functionality: a. User Interface Functions: Functions handle user interactions, allowing input of source code files and providing compiled output through a simple user interface or command-line interface.