COMSATS Institute of
Information Technology

Group Members:

M. Abdullah Arshad (SP20-BCS-033)

Hasnain Ahmed (FA20-BCS-005)

Class/Section: BCS-7 (A)

Subject: CC-Lab (Compiler Construction)

Submission To: Sir Bilal Haider

Date: 28-Dec-2023

CC-Lab Terminal:

Question 4: Explain how functions works step by step?

Answer:

❖Lexical Analyzer:

- **Form1 Class:** Form1 is a class that represents a Windows Forms application.

It contains an event handler btnAnalyze_Click associated with the button named btnAnalyze. When this button is clicked, it triggers the analysis of the input text.

- **btnAnalyze_Click Event Handler:** This function is called when the "Analyze" button is clicked.

It retrieves the input text from a TextBox named txtInput.

Calls the Analyze function with the input text and obtains a list of Token objects.

Clears the items in a ListBox named lstTokens.

Iterates through the list of tokens and adds each token to the ListBox for display.

- **Analyze Function:** Analyze function takes a string input and returns a list of Token objects.

Initializes a list called tokens to store the identified tokens.

Defines regular expressions for various token types, such as keywords, identifiers, numbers, string literals, operators, punctuation, comments, and whitespaces.

Removes comments and whitespaces from the input string using regular expressions.

Tokenizes the input string based on the defined regular expressions and creates Token objects for each match.

The created tokens are added to the tokens list.

Finally, the tokens list is sorted based on the token positions in the input string and returned.

- **Token Class:** Token is a class representing a lexical token.

It has properties for Lexeme (the actual text of the token), Type (the type of the token), and Position (the position of the token in the input string).

A constructor is defined to initialize these properties when creating a new token.

The ToString method is overridden to provide a custom string representation of a token.

- **TokenType Enumeration:** An enumeration defining different types of tokens like Keyword, Identifier, Number, etc.

❖LR Parsing:

❖ **import_grammar(fileHandle) Function:** Reads a context-free grammar from a file and returns the grammar, terminals, and non-terminals.

G: List of productions.

T: List of terminals.

Nt: List of non-terminals.

It parses each line of the file to extract productions and builds the lists.

- **closure(I, G, Nt) Function:** Takes a set of LR(0) items I, the grammar G, and non-terminals Nt.

Computes the closure of the given set of LR(0) items by repeatedly expanding items until no more items can be added.

- **goto(I, X, Nt) Function:** Computes the GOTO set for LR(0) items.

Given a set of LR(0) items I, a symbol X, and non-terminals Nt, it calculates the set of items that can be obtained by shifting the dot over X.

- **items(G, T, Nt) Function:** Computes the LR(0) items for the grammar.

Returns the set of LR(0) items, the action table, goto table, reduction states, and the accept state.

It uses closure and goto to compute LR(0) items and transitions.

- **parse_input_string(G, T, Nt, action_list, goto_list, reduction_states, accept_state, input_str) Function:** Implements the LR(0) parsing algorithm for the given grammar and input string.

It initializes a stack and processes the input string based on the action and goto tables.

Outputs the parsing actions (shift, reduce, accept) and the production rules applied.

- **Main Execution Section (__main__):** Reads a grammar from a file specified by the user.

Calls import_grammar to get the grammar, terminals, and non-terminals.

Calls items to compute LR(0) items and related information.

Prints the terminals, non-terminals, action table, goto table, reduction states, and the accept state.

Asks the user to input a string for parsing.

Calls parse_input_string to parse the input string using the generated LR(0) parser.

-End