CSE341 Programming Languages (Fall 2023) Homework #3 Documentation

Yacc Interpreter Code

1. Definitions and Function Prototypes

- Standard Libraries: Inclusion of standard C libraries (stdio.h, string.h, stdlib.h).
- Fraction Operations: Functions for basic arithmetic operations (addFractions, subtractFractions,multiplyFractions, divideFractions) on fractions represented as strings.
- GCD and Fraction Reduction: Utilities to calculate the Greatest Common Divisor (calculateGCD) and to reduce fractions to their simplest form (reduceFraction).
- Structures for Variables and Functions: Structure 'variable' holds name-value pairs for variables.Structure 'function' defines function names and expression types. Arrays 'variables' and 'functions' to store multiple variables and functions.
- Symbol Table Functions: Functions (insertVariable, findVariableValue, updateVariableValue) tomanage a symbol table for variables.

2. Parser Rules (Yacc/Lex Syntax)

- Token Definitions: Definition of tokens for parsing, including keywords (like KW_EXIT, KW_DEF),operators (like OP_PLUS, OP_DIV), and types (IDENTIFIER, VALUEF).
- Parsing Rules: Expression Handling (EXP): Rules to handle arithmetic operations on expressions,which can be fractions or identifiers. It includes parsing, variable substitution, and invoking the appropriate arithmetic functions. Function Definition and Invocation (FUNCTION): Rules

for defining and invoking functions. Functions can have arithmetic expressions and use previously defined variables. Variable Setting (SET): Rule to assign values to variables.

3. Main Program and Utility Functions

- Main Function (main): Entry point of the program, calls yyparse() to start parsing.
- Error Handling (yyerror): Function to handle parsing errors.
- Fraction Arithmetic Implementations: Implementations of addFractions, subtractFractions,multiplyFractions, divideFractions, performing arithmetic on fractions and simplifying the results.
- Symbol Table Management: Implementations of insertVariable, findVariableValue, updateVariableValue for managing the symbol table.

4. Key Features

- Fraction Arithmetic: Performs addition, subtraction, multiplication, and division of fractions.
- Function Definitions and Scoping: Supports defining functions with arithmetic expressions.
- Variable Management: Maintains a symbol table for variables with scope handling.
- Parser Integration: Designed to work with yacc/lex for parsing expressions, function definitions,and variable assignments.

Lisp Interpreter Code

1. Main Interpreter Function

Function: gppinterpreter

Description: Initializes the interpreter. It chooses to start from a file or shell based on the provided argument.

2. File and Shell Processing Functions

Function: start-interpreter-from-file

Description: Reads and evaluates a file line by line, skipping comments.

Function: start-interpreter-from-shell

Description: Reads and evaluates input from the terminal until an empty string is entered.

3. Lexical Analysis and Tokenization

Function: tokenize-one-line

Description: Splits a line into tokens for processing.

Functions: start-DFA-part1, start-DFA-part2

Description: Perform DFA analysis on tokens, classifying them appropriately.

4. Operator and Keyword Handling

Functions: is-operator, is-comment, is-keyword, etc.

Description: These functions determine the type of each token (operator, comment, keyword, valuef, identifier).

5. Arithmetic Operations

Functions: resolve-plus, resolve-minus, resolve-div, resolve-mult Description: Perform arithmetic operations on fractional values.

6. File Writing and Syntax Error Handling

Functions: open-file-to-write, write-lexical-result-to-file, etc.

Description: Handle file writing operations and report any syntax errors.

7. Helper Functions

Functions: str-split, str-remove, delimiterp

Description: Utility functions for string manipulation and processing.

8. Program Execution

Function: start-tokenization

Description: Determines how to start the interpreter based on command line arguments.

9. Key Features and Usage Scenario

Key Features: File and Shell Input, Lexical Analysis, Arithmetic on Fractions, Syntax Error Handling, Extensibility.