**Compiler Construction Lab CS - 471L**

**Lab Manual – Week 1**

**Problem Statement:** Expression Evaluator with Step-by-Step Evaluation

You are tasked with writing a complete program in the language of your choice to evaluate mathematical expressions provided as input from the command line. The program must be capable of handling the following operators within expressions: +, -, \*, /, (), {}, and [].

**Program Requirements:**

**Input:** Your program should accept a single mathematical expression as a command-line argument. The expression can include the following:

* Numeric constants (e.g., 2, 3.14)
* Binary operators: +, -, \*, /
* Parentheses: ()
* Curly braces: {} for grouping
* Square brackets: [] for grouping

**Expression Evaluation:** The program should correctly evaluate the input expression according to the following rules and formats for Infix, Postfix and Prefix expressions:

* Respect operator precedence.
* Support the use of parentheses {}, and square brackets [] for altering the order of operations.
* Perform arithmetic operations accurately.
* Step-by-Step Evaluation: The program must provide a detailed step-by-step evaluation of the expression, including:
* Display each token (numbers, operators, and brackets) as it is processed.
* Show the intermediate results and applied operators at each step of the evaluation.

**Output**: After the evaluation, display the result of the expression.

**Examples**:

**Infix Expression:** Suppose the input infix expression is "3 \* (2 + 4) - [5 - 2]". The program should produce the following output:

|  |  |
| --- | --- |
| Expression: "3 \* (2 + 4) - [5 - 2]" Token: 3 Token: \* Token: ( Token: 2 Token: + Token: 4 Token: ) Applied operator: + Left: 2 Right: 4 Result: 6 Token: - Token: [ | Token: 5 Token: -  Token: 2 Token: ] Applied operator: - Left: 5 Right: 2 Result: 3 Applied operator: \* Left: 3 Right: 6 Result: 18 Applied operator: - Left: 18 Right: 3 Result: 15 Result: 15 |

**Postfix Expression:**

Suppose the input postfix expression is "2 3 4 + \* 5 2 - [] -". The program should produce the following output:

|  |  |
| --- | --- |
| Expression: "2 3 4 + \* 5 2 - [] -" Token: 2 Token: 3 Token: 4 Token: + Applied operator: + Left: 3 Right: 4 Result: 7 Token: \* Applied operator: \* Left: 2 Right: 7 Result: 14 | Token: 5  Token: 2 Token: - Applied operator: - Left: 5 Right: 2 Result: 3 Token: [] Applied operator: - Left: 14 Right: 3 Result: 11 Result: 11 |

**Prefix Expression:**

Suppose the input prefix expression is "- \* 3 + 2 4 - [] 5 2". The program should produce the following output:

|  |  |
| --- | --- |
| Expression: "- \* 3 + 2 4 - [] 5 2" Token: - Token: \* Token: 3 Token: + Token: 2 Token: 4 Applied operator: + Left: 2 Right: 4 Result: 6 Token: - | Token: [] Token: 5  Token: 2 Applied operator: - Left: 5 Right: 2 Result: 3 Applied operator: \* Left: 3 Right: 6 Result: 18 Applied operator: - Left: 18 Right: 3 Result: 15 Result: 15 |

**Constraints:**

* The input expression will be a valid mathematical expression containing the specified operators.
* The expression may contain both integers and floating-point numbers.
* The expression length will not exceed a reasonable limit.

**Submission:**

Please submit the complete program code along with any necessary documentation on how to compile and run the program. Ensure that your program adheres to the requirements and produces the expected output.