Expression Evaluator

Part one:

Implement a C++ program that converts infix expressions to both prefix and postfix notation. Infix expressions are the most commonly used format in mathematics and computer science, where operators are placed between operands (e.g., 2 + 3 * 4). However, prefix (Polish) and postfix (Reverse Polish) notations are often preferred for simplicity and efficiency in parsing and evaluating expressions.

Your program should take an infix expression as input and provide two output options:

- Convert the infix expression to its equivalent prefix notation.
- Convert the infix expression to its equivalent postfix notation.

The program should handle expressions that include the following operators: addition (+), subtraction (-), multiplication (*), division (/), and parentheses for grouping. The precedence of operators should be correctly maintained (e.g., follow the standard order of operations).

Example Input and Output:

Input: "(2 + 3) * 4"	Output: Prefix: "* + 2 3 4"	Postfix: "2 3 + 4 *"
Input: "5 * (6 + 2) / 4"	Output: Prefix: "/ * 5 + 6 2 4"	Postfix: "5 6 2 + * 4 /"

Part two:

Implement a C++ program that evaluates both prefix and postfix expressions. Prefix and postfix notations are efficient and stack-based ways to represent mathematical expressions, and the program should be able to correctly calculate the result of such expressions.

Your program should offer the following functionalities:

- Evaluate a given prefix expression and compute its result.
- Evaluate a given postfix expression and compute its result.

The program should handle expressions that include the following operators: addition (+), subtraction (-), multiplication (*), division (/), and parentheses for grouping. The precedence of operators should be correctly maintained (e.g., follow the standard order of operations).

Example Input and Output:

Input (Prefix): "+ * 2 3 4" Output: "Result: 10"

Input (Postfix): "2 3 * 4 +" Output: "Result: 10"

Input (Prefix): "/ * 5 + 6 2 4" Output: "Result: 10"

Input (Postfix): "5 6 2 + * 4 /" Output: "Result: 10"