Documentation for Evaluating Reverse Polish Notation

Problem Description

You are given an array of strings named tokens that represents an arithmetic expression in Reverse Polish Notation (RPN). Your task is to evaluate this expression and return the integer result.

Definition

Reverse Polish Notation (RPN) is a mathematical notation where every operator follows all of its operands. For example, the expression "3 4 +" means "3 + 4".

Key Points

1. **Operators:**
   - *The valid operators in RPN are:*
     - + (Addition)
     - (Subtraction)
     - (Multiplication)
     - / (Division)

2. **Operands:**
   - *Each operand can be either:*
     - An integer value (e.g., "2", "-3")
     - Another RPN expression (which will be evaluated recursively)

3. Division:

- The division operation truncates towards zero. For example, -1 / 2 will result in 0, and 7 / -3 will result in -2.

4. Expression Validity:

- The input will always represent a valid arithmetic expression in RPN.
- Division by zero is not a concern, as it will not occur.

5. Integer Range:

- The result of the evaluation, as well as all intermediate calculations, will fit within a 32-bit integer range.

Function Signature

```
def evalRPN(tokens: List[str]) -> int:
    pass
```

Parameters

- **tokens:** A list of strings representing the RPN expression. Each element is either an operator (+, -, *, /) or an integer (within the range [-200, 200]).

Returns

- An integer representing the result of the evaluated RPN expression.

1. **Example 1**
    - **Input:** ["2", "1", "+", "3", "*"]
    - **Output:** 9
    - **Explanation:** The RPN expression ["2", "1", "+", "3", "*"] is evaluated as ((2 + 1) * 3), which results in 9.

2. **Example 2**

- **Input:** ["4", "13", "5", "/", "+"]
- **Output:** 6
- **Explanation:** The RPN expression ["4", "13", "5", "/", "+"] is evaluated as (4 + (13 / 5)), which results in 6.

3. **Example 3**

- **Input:** ["10", "6", "9", "3", "+", "-11", "*", "/", "*", "17", "+", "5", "+"]
- **Output:** 22
- **Explanation:** *The RPN expression ["10", "6", "9", "3", "+", "-11", "*", "/", "*", "17", "+", "5", "+"] is evaluated step-by-step as follows:*
    - Compute 9 + 3 = 12
    - Compute 12 * -11 = -132
    - Compute 6 / -132 = 0
    - Compute 10 * 0 = 0
    - Compute 0 + 17 = 17
    - Compute 17 + 5 = 22
    - Final result: 22

Constraints

- 1 <= tokens.length <= 10,000
- Each tokens[i] is either an operator (+, -, *, /) or an integer in the range [-200, 200].