Problem 24: RPN

Difficulty: Hard

Originally Published: Code Quest 2015

Problem Background

Reverse Polish Notation (RPN, also called postfix notation) is a mathematical notation where the operator (+, -, *, /, etc.) follows the operands. For example, 2 + 3 in postfix notation is 2 3 +. This notation has the advantage that it's easy to evaluate in a computer program and it doesn't require parenthesis to handle order of operations.

Problem Description

Your task is to write a program that will convert expressions from algebraic (also called infix) notation to RPN.

Some Notes on RPN

- Operators use the two operands to their left if you're looking at the final RPN equation. Once used, the result becomes available as an operand for another operation.
  - A + B – C * D --> A B + C D * -
    In this example, A and B are added together and saved for later. C and D are multiplied together and saved for later. The minus sign at the end acts on both saved values and subtracts the two.
- Operators are used from left to right. Note the difference between these two lines because we know multiplication has to happen before addition. The order of the operators is switched in the second example to account for the order in which the operations must happen.
  - A * B + C --> A B * C +
  - A + B * C --> A B C * +
- Exponentiation is right-associative (meaning if two exponent operations are next to each other, they happen from right to left instead of left to right):
  - 2 ^ 2 ^ 3 is the same as 2 ^ (2 ^ 3), not (2 ^ 2) ^ 3
- The operands should stay in the same order in your output. For example, if you were given the expression A + B, the answers A B + and B A + are mathematically equivalent because of the commutative property of addition. However, for the purposes of this problem, the only acceptable answer would be A B +.

Sample Input

The first line of your program's input, received from the standard input channel, will contain a positive integer representing the number of test cases. Each test case will include:

- A positive number **N** representing the number of regular algebraic notation for you to convert.
- **N** lines, containing one an expression in regular algebraic notation for you to convert. Each character in the expression will be separated by a space.
- The only non-operand characters you will encounter are PEMDAS characters:
    - (and) for grouping
    - ^ for exponentiation
    - * and / for multiplication and division
    - + and – for addition and subtraction
- If you are unfamiliar with PEMDAS, it is a pneumonic device for remembering the order in which operations are performed:
    - Parentheses first (innermost to outermost)
    - Exponentiation second
    - Multiplication and Division next (they have the same precedence)
    - Addition and Subtraction last (they have the same precedence)

```
2
2
a * b + c
2 ^ 2 ^ 3
3
( a - b ) * c / d + e / f
( ( a + b ) - ( c - d ) ^ x ) + q * w
A + B * C - D
```

Sample Output

Your program should convert the expressions in the order they are read to RPN. All your output characters should be separated by a single space.

```
a b * c +
2 2 3 ^ ^
a b - c * d / e f / +
a b + c d - x ^ - q w * +
A B C * + D -
```