[ED005] Notação Polaca Invertida

Neste problema deverá submeter uma classe **ED005** contendo um programa completo para resolver o problema (ou seja, com o método main).
Pode assumir que no Mooshak terá acesso às classes de listas, pilhas e filas como dadas nas aulas (ou seja, não precisa de as incluir no código submetido).

O problema

Quando andaste na escola primária, aprendeste que os diferentes operadores aritméticos têm diferentes precedências. Isso acontece em todos os países, e por exemplo os ingleses usam a mnemónica: *"**P**lease **E**xcuse **M**y **D**ear **A**unt **S**ally"*, que indica a ordem dos operadores (**P**arenteses, **E**xponenciação, **M**ultiplicação, **D**ivisão, **A**dição, **S**ubtração).

Por exemplo, a expressão:

```
1 + 2 x 3 = ?
```

dá como resultado `7`, pois o operador multiplicação tem precedência.

Se quiseremos avaliar primeiro a soma a expressão tem de ser:

```
(1 + 2) x 3 = ?
```

No entanto, nos tempos iniciais das calculadoras electrónicas, revelou-se complicado analisar este tipo de expressões.

Por isso mesmo, em 1920, o matemático polaco **Jan Lukasiewicz** criou uma notação que tornava desnecessário o uso de parenteses, garantindo sempre que as operações eram efectuadas como desejado.
A notação consistia basicamente em escrever os operadores antes dos números e não no meio deles.

Já em 1950, o *computer scientist* **Charles L. Hamblin** propôs um esquema onde os operadores apareciam a seguir aos números, em vez de ser antes. Esta notação acbou por ser muito usada, devido entre outras coisas à sua fácil implementação num sistema automático usando uma pilha, e ficou conhecida como **Notação Polaca Invertida (RPN)** (ou *postfix*). Em RPN, uma expressão arbitrariamente complexa pode ser escrita sem o uso de parenteses.

Notação Normal	**Notação Polaca Invertida (RPN)**
1 + 1	1 1 +
3 * (4 + 5)	3 4 5 + *
3 + 4 * 5	3 4 5 * +

A tua tarefa é criar um programa capaz de calcular o valor final de expressões dadas em RPN, sabendo que um algoritmo para as analisar é:

- Criar uma pilha
- Ir da esquerda para a direita lendo elemento a elemento e:
  - Se o elemento for um número, inseri-lo na pilha
  - Se for um operador, retirar dois valores da pilha, aplicar esse operador, e inserir o resultado na pilha
- No final, fica apenas um valor na pilha, que é o resultado da expressão
- Se a pilha não corresponder a alguma das coisas que foi dita atrás (por exemplo, ficar vazia), então a expressão é incorrecta.

Input

A primeira linha contém um número **N**, indicando o número de expressões a analisar.

As seguintes **N** linhas contêm expressões RPN, contendo

- Dígitos, representando números inteiro
- Caracteres, representando operações (+, -, * , /)

Podes assumir que os números são sempre inteiros, que os cálculos intermédios vão dar sempre números inteiros, e que só aparecem os quatro tipos básicos de operações atrás citados.

Output

Uma linha para cada expressão, indicando o resultado final caso a expressão seja correcta, ou **`Expressao Incorrecta`**, caso contrário.

Exemplo de input/output

Input	Output
6 1 1 + 3 4 5 + * 3 4 5 * + 1 + 1 1 1 + 2 3 8 2 / - 1 + *	2 27 23 Expressao Incorrecta Expressao Incorrecta 0

Última actualização: **14:12, 17/09/2004**