William works as a bookkeeper at a factory that produces calculators. His job requires he create arithmetic expressions for various important bookkeeping activities. William is often annoyed with having to manage the order of operations and parentheses in his expressions and often wonders if there's a better way.

One day when eating lunch, William overhears two software engineers discussing the benefits of postfix notation and how it doesn't require parentheses. Unfortunately, all William hears about this form of notation is that operators are always written after operands. William returns from lunch and begins recording expressions with this in mind.

On one page William records the next operand in his expression and on another the writes next operator. To save space, if there exists multiple sequential operators William records that operator and the number of times to apply it sequentially. William evaluates his expressions by evaluating the leftmost operations first, applying the leftmost operator to the leftmost operands.

Example:

```
1 2 5 4 3 6 8
+3-/*
```

```
((1+2+5+4-3)/6)*8 = 12
```

After a week of this practice, William was fired. Your team was hired by William's previous employer to write a program that evaluates William's notation.

Input

The first line of input will contain a single integer **T**, followed by **T** expressions. Each expression consists of three lines, the first consisting of an integer **N**, the second contains **N** space separated integers, and the third contains instructions for **N-1** operations `(+|-|*|/)[2-9]?`.

Output

For each expression print the value that the expression evaluates to. All expressions will evaluate to integers.

Sample Input	Sample Output
2 7 1 2 3 4 6 5 4 +-3/* 14 10 5 23 3 9 3 49 19 5 23 24 2 3 5 +2*-/+-/+2/3	-8 2