Given a sequence of integers, $a_1, a_2, \ldots, a_n$, we define its *sign matrix* S such that, for $1 \le i \le j \le n$, $S_{ij} =$ "+" if $a_i + \ldots + a_j > 0$; $S_{ij} =$ "−" if $a_i + \ldots + a_j < 0$; and $S_{ij} =$ "0" otherwise.

For example, if $(a_1, a_2, a_3, a_4) = (-1, 5, -4, 2)$, then its sign matrix S is a 4×4 matrix:

$$\begin{array}{ccccc} & 1 & 2 & 3 & 4 \\ 1 & - & + & 0 & + \\ 2 & & + & + & + \\ 3 & & & - & - \\ 4 & & & & + \end{array}$$

We say that the sequence $(-1, 5, -4, 2)$ *generates* the sign matrix. A sign matrix is *valid* if it can be generated by a sequence of integers.

Given a sequence of integers, it is easy to compute its sign matrix. This problem is about the opposite direction: Given a valid sign matrix, find a sequence of integers that generates the sign matrix. *Note that two or more different sequences of integers can generate the same sign matrix.* For example, the sequence $(-2, 5, -3, 1)$ generates the same sign matrix as the sequence $(-1, 5, -4, 2)$.

Write a program that, given a *valid* sign matrix, can find a sequence of integers that generates the sign matrix. You may assume that every integer in a sequence is between -10 and 10, both inclusive.

Input

The input consists of T test cases. The number of test cases T is given in the first line of the input. Each test case consists of two lines. The first line contains an integer n ($1 \le n \le 10$), where n is the length of a sequence of integers. The second line contains a string of $n(n+1)/2$ characters such that the first n characters correspond to the first row of the sign matrix, the next $n-1$ characters to the second row, ..., and the last character to the n-th row.

Output

For each test case, output exactly one line containing a sequence of n integers which generates the sign matrix. If more than one sequence generates the sign matrix, you may output any one of them. Every integer in the sequence must be between -10 and 10, both inclusive.

Sample Input

```
3
4
-+0++++--+
2
+++
5
++0+-+-+--+-+--
```

Sample Output

```
-2 5 -3 1
3 4
1 2 -3 4 -5
```