University Homework

Hugo had quite a bad day today. His favourite university subject, mathematical analysis, was taught by his least liked substitute teacher - he always gave out an abnormaly large amount of homework. Today was no different.

The teacher wrote **N** numbers on the board, and after a long, thoughtful pause he told the class with a grin: "Well then, kids. For today's homework you will do this harmless exercise. As you can see, I've written quite a few numbers on the board, and your task is to find which smallest non-empty subset of them has the greatest product. That should keep you busy for today's evening!".

Sigh.
How could someone even come up with such a boring, time-consuming and impractical task. If only there was someone who would help Hugo and do it for him.

Input

The first line of input contains the number **$1 \le T \le 15$**: the number of test cases. **T** cases follow.
The first line of each case contains the number **$1 \le N \le 10^5$**: how many numbers the teacher wrote on the board. The second line contains **N** integers; their absolute value will not exceed **10^9**.
You may assume that in any input file, the sum of **N** across all test cases does not exceed **$3*10^5$**.

Output

For each test case, first print the string 'Case x: ', where x is the number of the test case, starting with 1. Then print a binary string (consistring of **0**'s and **1**'s) ; the **i^{th}** character should be **1** if you decided to include the **i^{th}** number from the input in the subset.

To make the output unambiguous, it should have the following properties:

1. It should contain **N** characters, at least one of which has to be a **1**.
2. If we multiply the numbers from the input at whose indices this string has the character **1**, the result will be the greatest possible.
3. Out of all binary strings with the above properties, it should contain the minimum number of **1**'s.
4. Out of all binary strings with the above properties, it should be lexicographically largest.

Reminder: To compare two valid strings lexicographically, find the first position from the left at which they differ. The one with a **1** at this position is lexicographically larger.

Example

Input:

```
1
6
-2 -2 -3 4 5 1
```

Output:

```
Case 1: 101110
```

The product of the subset this string represents is **(-2) * (-3) * 4 * 5 = 120**, which is the greatest possible. It uses the first -2 from the input, as using the second one would result in a lexicographically smaller string. It does not use the 1, as the product would remain the same but the number of **1**'s in the string would increase.