Your friend, a biochemistry major, tripped while carrying a tray of computer files through the lab. All of the files fell to the ground and broke. Your friend picked up all the file fragments and called you to ask for help putting them back together again.

Fortunately, all of the files on the tray were identical, all of them broke into exactly two fragments, and all of the file fragments were found. Unfortunately, the files didn't all break in the same place, and the fragments were completely mixed up by their fall to the floor.

You've translated the original binary fragments into strings of ASCII `1`'s and `0`'s, and you're planning to write a program to determine the bit pattern the files contained.

Input

?The input begins with a single positive integer on a line by itself indicating the number of the cases following, each of them as described below. This line is followed by a blank line, and there is also a blank line between two consecutive inputs.

Input will consist of a sequence of "file fragments", one per line, terminated by the end-of-file marker. Each fragment consists of a string of ASCII `1`'s and `0`'s.

Output

For each test case, the output must follow the description below. The outputs of two consecutive cases will be separated by a blank line.

Output is a single line of ASCII `1`'s and `0`'s giving the bit pattern of the original files. If there are $2N$ fragments in the input, it should be possible to concatenate these fragments together in pairs to make N copies of the output string. If there is no unique solution, any of the possible solutions may be output.

Your friend is certain that there were no more than 144 files on the tray, and that the files were all less than 256 bytes in size.

Sample Input

```
1

011
0111
01110
111
0111
10111
```

Sample Output

```
01110111
```