Problem D. Go++

Problem

The Go language was designed to have a simple API and to support multi-threading. The Code Jam team wants to push these goals to the limit, so we are proposing a new language called Go++.

The Go++ language uses one register, which stores one boolean value (0 or 1). This register is initialized to 0. The language has three instructions:

- `0`, which sets the register to 0.
- `1`, which sets the register to 1.
- `?`, which prints the current register value.

Simple, right? To support multi-threading, we allow two different Go++ programs to run simultaneously while sharing the one register. Each instruction executes atomically — that is, one instruction must completely finish before the next instruction can start. However, the two programs may be interleaved in any way that preserves the relative order within each program.

For example, here are the only six ways in which the two programs `1?` and `?0` could be executed together. (The underline on the second program is just to distinguish its instructions from the instructions in the first program.)

- `?01?`, which will print `01`. (Remember that the register is initialized to 0.)
- `?10?`, which will print `00`.
- `?1?0`, which will print `01`.
- `1?0?`, which will print `10`.
- `1??0`, which will print `11`.
- `1??0`, which will print `11`.

Note that the output string always consists of `0`s and `1`s, and never `?`s, since `?` is not a state the register can be in.

Usually, programmers write programs to produce a desired output, but your task will be to write two programs that *won't* produce an *undesired* output! Specifically, you will be given a "bad" string **B** of length **L**, and a set **G** of **N** "good" strings, all of length **L**. You must produce two Go++ programs (not necessarily of the same length), which, when run in the way described here, could produce *all* of the strings in **G**, but could *not* produce the string **B**. It is fine if the programs could also produce other strings that are not **B** and not in **G**. Note that there must be a combined total of exactly **L** `?` instructions in the two programs. The combined number of instructions in the two programs must not exceed 200.

For example, for **B** = `11` and **G** = { `10`, `00` }, the programs `?` and `10?1` would be one valid answer. They can produce every string in **G**, but they cannot produce **B**, no matter how they are interleaved. (They can also produce the string `01`, which is not **B** and is not in **G**, but that is fine.) However, the programs `1?` and `?0` would not be a valid answer, since (as we saw above) they can produce **B**. The programs `00` and `??` would not be a valid answer, since they cannot produce every string in **G**.

Can you produce two programs that satisfy the conditions, or determine that the task is `IMPOSSIBLE`?

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow; each consists of three lines. The first line of each test case has two integers **N** and **L**: the number of strings in **G**, and the length of the **B** string and the strings in **G**. The second line has **N** different strings of length **L**: the strings in **G**. The third line has one string of length **L**: the bad string **B**. **B** and all of the strings in **G** are made up of only `0`s and/or `1`s.

Output

For each test case, output one line containing `Case #x: IMPOSSIBLE`, if no programs will satisfy the conditions; otherwise, output `Case #x: y z`, where x is the test case number (starting from 1) and y and z are your two programs that satisfy the conditions. The combined number of instructions in your programs must not exceed 200. Each program must contain at least one instruction. There must be a combined total of exactly **L** `?` instructions in the two programs. If there are multiple correct outputs, print any of them.

Limits

$1 \le \mathbf{T} \le 100$.
$1 \le \mathbf{N} \le 100$.
$1 \le \mathbf{L} \le 50$.
All strings in **G** are different.

Small dataset

B consists entirely of 1s.

Large dataset

B may be any string consisting of 0s and/or 1s.

Sample

Input	Output
3 2 2 10 00 11 3 2 11 10 00 01 4 2 00 01 10 11 11	Case #1: ? 10?1 Case #2: 1?? 0 Case #3: IMPOSSIBLE

The sample output displays one set of answers to the sample cases. Other answers may be possible.

Sample case #1 is the one described in the problem statement.

Sample case #2 would not appear in the Small dataset.

Sample case #3 is obviously `IMPOSSIBLE` because **B** is in **G**.