Input file:	**parentrises.in**
Output file:	**parentrises.out**
Time limit:	**0.3 second**
Memory limit:	**256 megabytes**

We define a string as a sequence of characters `'('` and `')'`.

We define a well bracketed string as a string that can be transformed into a correct arithmetic expression by inserting characters `'1'` and `'+'` between the characters of the string. For example, strings `"()()"` and `"(())"` are correct and their possible expressions are `"(1)+(1)"` and `"((1+1)+1)"` respectively, whilst `")("` and `"("` are not. The empty string is defined as well bracketed.

Bogdan Shepherd and Rolling Costel work in the B.U.A.P. (Bureau of Unnecessary Algorithmic Problems), useless brackets division. They usually receive two types of tasks:

1. Theo of Fire and Radu the Mountain (their clients), both of which are colorblind, have a string `S`. Theo can only see Red and Green colors, whereas Radu can only see Blue and Green colors. They want to color `S` such that both of them will see a well bracketed string if they each ignore the brackets with colors they can't see, using only the three available colors: Red, Green and Blue. If such a coloring exists, we say that `S` is **RGB-readable**. Find a possible coloring or report that none exist.
2. Mihai Likelance and Andrei the Priest (other clients) want to know what is the number of different **RGB-readable** strings of length `N`, for a given `N`. They want to find the answer modulo `1.000.000.007`.

Bogdan and Costel usually solve the two questions using C / C++, but the Grandpa of Evil hacked their computers and they can only use Rust. Unfortunately they don't know anything about Rust; can you help them? Of course, you do not have such limitations, so you can use any language you want (i.e. C / C++ / Pascal).

Input

The first line contains an integer `P`, denoting the type of problem you will have to solve:

- If `P = 1`, you will solve Theo's and Radu's requests.
- If `P = 2`, you will solve Mihai and Andrei's requests.

The second line contains an integer `T`, denoting the number of test cases in the file.
Each of the following `T` lines will contain a test case:

- If `P = 1`, you will receive a string `S`.
- If `P = 2`, you will receive a number `N`.

Output

For each test print the answer on a separate line.
If `P = 1` and there is no solution, print the text `"impossible"` (quotes for clarity).
If `P = 1` and there is a solution, print a possible coloring. The i-th character on the line should denote the color of the i-th character in `S`, encoded with `'R'` for red, `'B'` for blue and `'G'` for green.
If `P = 2`, print the answer modulo `1.000.000.007`.

Scoring

For `P = 1`:

Let `L` be the total number of characters of the strings in the input file, then:

Subtask 1 (5 points)
- `1 ≤ T ≤ 5`
- `1 ≤ length of S ≤ 13`

Subtask 2 (11 points)
- `1 ≤ L ≤ 100`

Subtask 3 (6 points)
- `1 ≤ L ≤ 1.000`

Subtask 4 (28 points)
- `1 ≤ L ≤ 1.000.000`

For `P = 2`:

Subtask 5 (6 points)
- `1 ≤ N, T ≤ 15`

Subtask 6 (16 points)
- `1 ≤ N, T ≤ 30`

Subtask 7 (28 points)
- `1 ≤ N, T ≤ 300`

Examples

parentrises.in	parentrises.out
1 3 ())(() ()(() ())))	GRBRBG BBRBG impossible

Notes

In the first case, both Theo and Radu will see the string "`()()`".
In the second case, Theo will see the string "`()`" and Radu will see the string "`()()`".
In the third case, there is no valid coloring.

parentrises.in	parentrises.out
2 2 6 100	12 959772055