Problem 12: Brackets
14 Points

Problem ID: `brackets`
Rank: `4`

Introduction

Big Ben, after working at McAlico's, was made fun of by all of his friends and became sad, so he looked for solace in brackets. After staring soulfully into the (metaphorical) eyes of the brackets, he invented something he called "Bear Brackets" (because they hug each other like a bear). Hey, everyone deserves some love...even brackets and Big Ben (and you!).

Problem Statement

A Bear Bracket is a sequence of brackets that follows these properties:

- An empty sequence of brackets is a Bear Bracket.
- If A is a Bear Bracket, then adding the proper brackets to both sides of A, ie. `(`A`)`, is also a Bear Bracket.
- If A and B are Bear Brackets, then AB is also a Bear Bracket.

For example, the following are valid Bear Brackets: "`(()) ()`", "`()`". These are not valid Bear Brackets: "`) (`", "`((`", and "`) () (`".

However, this is not where Big Ben's invention ends. He found a way to deconstruct these Bear Brackets! Big Ben writes a "0" for an open bracket "(" and writes a "1" for a closed bracket ")". For example, the Bear Bracket "`() (())`" is written into the binary string `010011`.

Given a binary string $\mathbf{X}$ of even length $\mathbf{N}$, output the minimum number, k, of non-empty balanced Bear Brackets required to reconstruct the original binary string when each of their strings are combined using the $\oplus$ (XOR) operation.

Another way to state this: if $f(X)$ is the operation that converts a bracket sequence X (not necessarily a Bear Bracket) into a binary string, where string X has even length, then the output needs to be the minimum $k \geq 1$ such that there exist Bear Brackets $S_1, S_2, S_3, \ldots S_k$ for which:

$$f(\mathbf{X}) = f(S_1) \oplus f(S_2) \oplus f(S_3) \oplus \cdots \oplus f(S_k).$$

Note that it is sometimes not possible to find any k that work. If that is the case, then output "No".

Also note that $1 \oplus 1 = 1$, $0 \oplus 1 = 0$, $1 \oplus 0 = 0$, and $0 \oplus 0 = 0$. For two digit or more calculations, the signs carry over. For example, $01 \oplus 11 = 10$.

Input Format

The first line of the input contains a single integer $\mathbf{T}$ denoting the number of test cases that follow.

For each test case:

- The first line contains a single integer $\mathbf{N}$ denoting the length of the binary string $\mathbf{X}$.
- The second line contains the single binary string $\mathbf{X}$.

Output Format

For each test case, if it is impossible to find Bear Brackets $S_1, S_2, \ldots, S_k$, output a single line containing the string No

Otherwise, your output should have multiple lines:

- The first line should contain a single integer k denoting the minimum number of Bear Brackets utilized.
- The next k lines describe all the Bear Brackets utilized. The i-th line of these lines contains a Bear Bracket of length $\mathbf{N}$, containing only (and), indicating the balanced Bear Bracket sequence S_i.

If there are multiple possible solutions, you may print any one of them.

Constraints

$1 \leq \mathbf{T} \leq 10^5$

$1 \leq \mathbf{N} \leq 10^6$

$\mathbf{N}$ is always even.

$k \geq 1$

It is guaranteed that the sum of $\mathbf{N}$ over all test cases does not exceed 10^6.

It is guaranteed that the sum of $\mathbf{N}k$ over all test cases does not exceed 2×10^7.

Sample Test Cases

Sample Input Download

```
3
4
1101
4
0110
2
01
```

Sample Output Download

```
No
Yes
2
()()
(())
Yes
1
()
```

Sample Explanations

For the first test case, 1101 is the equivalent to $))()$. Therefore, we know that this itself is not a Bear Bracket. Trying to break it into cases to check $k = 2, k = 3, k = 4$, it also possible to show there are no brackets such that 1101 can be reconstructed.

The second test case 0110 is the equivalent to $())($. $k = 1$ doesn't work because 0110 itself is not a Bear Bracket. On the other hand, $()()$ is equal to 0101 and $(())$ is equal to 0011, which simplifies when $0101 \oplus 0011 = 0110$. Note that both of these sequences are Bear Brackets themselves.

The third test case is itself a Bear Bracket, so outputting $k = 1$ and the given binary representation into a bracket is all that is necessary.