Brackets Problem Code: **BRACKETS**

A valid parentheses sequence is a non-empty string where each character is either '**(**' or '**)**', which satisfies the following constraint:

You can find a way to repeat erasing adjacent pairs of parentheses '**()**' *until it becomes empty.*

For example, '**(())**' and '**()((()()))**' are valid parentheses sequences, but '**)()(**' and '**(()**' are not.

Mike has a valid parentheses sequence. He really likes everything about his sequence, except the fact that it is quite long. So Mike has recently decided that he will replace his parentheses sequence with a new one in the near future. But not every valid parentheses sequence will satisfy him. To help you understand his requirements we'll introduce the pseudocode of function **F(S)**:

```
FUNCTION F( S - a valid parentheses sequence )

BEGIN

        balance = 0

        max_balance = 0

        FOR index FROM 1 TO LENGTH(S)

        BEGIN

                if S[index] == '(' then balance = balance + 1

                if S[index] == ')' then balance = balance - 1

                max_balance = max( max_balance, balance )

        END

        RETURN max_balance

END
```

In other words, **F(S)** is equal to the maximal balance over all prefixes of **S**.

Let's denote **A** as Mike's current parentheses sequence, and **B** as a candidate for a new one. Mike is willing to replace **A** with **B** if **F(A)** is equal to **F(B)**. He would also like to choose **B** with the minimal possible length amongst ones satisfying the previous condition. If there are several such strings with the minimal possible length, then Mike will choose the least one lexicographically, considering '**(**' to be less than '**)**'.

Help Mike!

Input

The first line of the input contains one integer **T** denoting the number of testcases to process.

The only line of each testcase contains one string **A** denoting Mike's parentheses sequence. It is guaranteed that **A** only consists of the characters **'('** and **')'**. It is also guaranteed that **A** is a valid parentheses sequence.

Output

The output should contain exactly **T** lines, one line per each testcase in the order of their appearance. The only line of each testcase should contain one string **B** denoting the valid parentheses sequence that should be chosen by Mike to replace **A**.

Constraints

$1 \leq \mathbf{T} \leq 5$;

$1 \leq |\mathbf{A}| \leq 100000(10^5)$.

Example

Input:

```
1

()((()()))
```

Output:

```
((()))
```