Problem H
Necklace Decomposition

source: `necklace.c` *or* `necklace.cpp` *or* `necklace.java`

Description

The set of cyclic rotations of a string are the strings obtained by embedding the string clockwise on a ring, with the first character following on the last, starting at any character position and moving clockwise on the ring until the character preceeding the starting character is reached. A string is a necklace if it is the lexicographically smallest among all its cyclic rotations. For instance, for the string 01011 the cyclic rotations are (10110, 01101, 11010, 10101, 01011), and furthermore 01011 is the smallest string and hence, a necklace.

Any string S can be written in a unique way as a concatenation $S = T_1T_2 \dots T_k$ of necklaces T_i such that $T_{i+1} < T_i$ for all $i = 1, \dots, k-1$, and T_iT_{i+1} is not a necklace for any $i = 1, \dots, k-1$. This representation is called the necklace decomposition of the string$S$, and your task is to find it.

The relation $<$ on two strings is the lexicographical order and has the usual interpretation:$A < B$ if A is a proper prefix of$B$ or if A is equal to B in the first $j-1$ positions but smaller in the$j$th position for some j. For instance, 001 $<$ 0010 and 1101011 $<$ 1101100.

Input

On the first line of the input is a single positive integer$n$, telling the number of test scenarios to follow. Each scenario consists of one line containing a non-empty string of zeros and ones of length at most 100.

Output

For each scenario, output one line containing the necklace decomposition of the string. The necklaces should be written as '(' necklace ')'.

Sample

Input	Output

`5`
`0`
`0101`
`0001`
`0010`
`11101111011`
	`(0)`
	`(0101)`
	`(0001)`
	`(001)(0)`
	`(111)(01111)(011)`