Google Code Jam '20 Qualification Round Problem B - Nesting Depth

Time limit: 20.0s **Memory limit:** 1G

tl;dr: Given a string of digits S, insert a minimum number of opening and closing parentheses into it such that the resulting string is balanced and each digit d is inside exactly d pairs of matching parentheses.

Let the *nesting* of two parentheses within a string be the substring that occurs strictly between them. An opening parenthesis and a closing parenthesis that is further to its right are said to *match* if their nesting is empty, or if every parenthesis in their nesting matches with another parenthesis in their nesting. The *nesting depth* of a position p is the number of pairs of matching parentheses m such that p is included in the nesting of m.

For example, in the following strings, all digits match their nesting depth: `0((2)1)`, `(((3))1(2))`, `((((4))))`, `((2))((2))(1)`. The first three strings have minimum length among those that have the same digits in the same order, but the last one does not since `((22)1)` also has the digits `221` and is shorter.

Given a string of digits S, find another string S', comprised of parentheses and digits, such that:

- all parentheses in S' match some other parenthesis,
- removing any and all parentheses from S' results in S,
- each digit in S' is equal to its nesting depth, and
- S' is of minimum length.

Input Specification

The first line of the input gives the number of test cases, T. T lines follow. Each line represents a test case and contains only the string S.

Output Specification

For each test case, output one line containing `Case #x: y`, where x is the test case number (starting from 1) and y is the string S' defined above.

Limits

Time limit: 20 seconds per test set.

Memory limit: 1GB.

$1 \leq T \leq 100.$

$1 \leq \text{length of } S \leq 100.$

Test set 1

Each character in S is either `0` or `1`.

Test set 2

Each character in S is a decimal digit between `0` and `9`, inclusive.

Sample Input

```
4
0000
101
111000
1
```

Sample Output

```
Case #1: 0000
Case #2: (1)0(1)
Case #3: (111)000
Case #4: (1)
```

The strings `()0000()`, `(1)0(((()))1)` and `(1)(11)000` are not valid solutions to Sample Cases #1, #2 and #3, respectively, only because they are not of minimum length. In addition, `1)(` and `)(1` are not valid solutions to Sample Case #4 because they contain unmatched parentheses and the nesting depth is 0 at the position where there is a 1.

You can create sample inputs that are valid only for Test Set 2 by removing the parentheses from the example strings mentioned in the problem statement.