A string consisting of parenthesis '(' and ')' is called **balanced string if any of the following is true.**

a. If the string is empty.

b. If A and B are balanced string, AB is also balanced string.

c. If A is a balanced string, (A) is also a balanced string

Now from a balanced string we can generate a sequence of numbers. The sequence is generated as follows:

- Start scanning the string from left to right and after each scan **print number of open parenthesis '(' scanned so far - number of close parenthesis ')' scanned so far (please note the minus sign in between)**
- After the sequence is generated it is rearranged randomly.

For example, suppose you have a balanced string = "`( )( )`"
The sequence generated after the first step: `1 0 1 0`
Sequence (one of many) after the rearrangement in the second step: `0 1 0 1`

Your task is — given a sequence in random order, output the lexicographically smallest balanced string if one exists, otherwise output '`invalid`'.

Note: '(' is lexicographically smaller than ')'. Also remember the string you generate must produce any sequence following above mentioned two steps which can be rearranged to match the sequence given in the input.

Input

There will be T ($1 \leq T \leq 11$) test cases. For each test, the first line contains an integer N ($1 \leq N \leq 100000$) where N denotes the number of integers in the sequence. The second line contains N integers separated by spaces describing the sequence. You can assume all the inputs in the sequence will fit in 32 bit signed integer data type.

Output

For each case you have to output the balanced string or '`invalid`' along with case number. There should not be any intermediate spaces between brackets in the output. See the output format below for more details.

Sample Input

```
2
4
0 0 1 1
5
1 2 3 4 5
```

Sample Output

```
Case 1: ()()
Case 2: invalid
```