It is known that Sheffer stroke function (NOT-AND) can be used to construct any Boolean function. The truth table for this function is given below:

Truth table for Sheffer stroke function

x	y	x — y
0	0	1
0	1	1
1	0	1
1	1	0

Consider the problem of adding two binary numbers A and B, each containing N bits. The individual bits of A and B are numbered from 0 (the least significant) to $N-1$ (the most significant). The sum of A and B can always be represented by $N+1$ bits. Let's call most significant bit of the sum (bit number N) the **overflow** bit.

Your task is to construct a logical expression using the Sheffer stroke function that computes the value of the overflow bit for arbitrary values of A and B. Your expression shall be constructed according to the following rules:

1. Ai is an expression that denotes value of i-th bit of number A.
2. Bi is an expression that denotes value of i-th bit of number B.
3. $(x|y)$ is an expression that denotes the result of Sheffer stroke function for x and y, where x and y are expressions.

When writing the index, i, for bits in A and B, the index shall be written as a decimal number without leading zeros. For example, bit number 12 of A must be written as '`A12`'. The expression should be completely parenthesized (according to the 3rd rule). No blanks are allowed inside the expression.

Input

The first line of the input contains an integer indicating the number of test cases in the input. After that there is a blank line and the test cases separated by a blank line.

Each test case consists of a single integer N ($1 \leq N \leq 100$), on a line by itself.

Output

For each test case, write to the output file an expression for calculating overflow bit of the addition of two N-bit numbers A and B according to the rules given in the problem statement.

Print a blank line between test cases.

Note: The stroke symbol (`|`) is an ASCII character with code 124 (decimal).

Sample Input

```
1

2
```

Sample Output

```
((A1|B1)|(((A0|B0)|(A0|B0))|((A1|A1)|(B1|B1))))
```