

## 10455 Gray Code

All of you know about Gray Code. It is a number code where consecutive numbers are represented by binary patterns that differ in one bit position only. In the following 4 examples of 3-bit gray code are shown:

```

0 0 0  0 0 0  0 0 0  0 0 0
0 0 1  0 0 1  0 1 0  0 1 0
0 1 1  0 1 1  0 1 1  0 1 1
0 1 0  0 1 0  0 0 1  0 0 1
1 1 0  1 1 0  1 0 1  1 0 1
1 1 1  1 0 0  1 0 0  1 1 1
1 0 1  1 0 1  1 1 0  1 1 0
1 0 0  1 1 1  1 1 1  1 0 0

```

In this problem we will deal with a gray code generation logic. This logic will generate the  $n$ -bit gray code using the coding of  $(n - 1)$  bits. Lets formally define the rules :

- Each gray code has a starting bit pattern. Such as '0 0 0' or '1 0 1', etc.
- An  $n$ -bit gray code will have  $2^n$  rows and two consecutive rows will differ by only one bit.
- Each bit pattern will be present exactly once.
- Gray code for 1-bit is trivial. Start with a bit and invert it in the next row.
- To construct  $n$ -bit gray code keep any of the  $n$  bits fixed (either 0 or 1) for the first  $2^{(n-1)}$  rows and use  $(n - 1)$ -bit gray code (generated using this logic) for remaining  $(n - 1)$  bits. Then invert the fixed bit for the next  $2^{(n-1)}$  rows and also use  $(n - 1)$ -bit gray code for remaining  $(n - 1)$  bits whose bit pattern of the first row is the same as the bit pattern of the last row of previous  $2^{(n-1)}$  rows. For example 2-bit gray code starting with '00' may be:

```

00      00
01      10
11  Or  11
10      01

```

Similarly 2-bit gray code starting with '01' may be:

```

01      01
00      11
10  Or  10
11      00

```

If you observe carefully, you will see that the 3-bit gray codes given above are also constructed using this logic. Many such gray codes are possible for a particular starting bit pattern. We can order them from 1 to  $G(n)$  where  $G(n)$  denotes the number of such gray codes for  $n$ -bit. In our ordering scheme:

- 1st  $n$ -bit gray code has its leftmost bit fixed and it uses 1st  $(n - 1)$ -bit gray code for upper half and also 1st  $(n - 1)$ -bit gray code for lower half.
- $G(n - 1)$ -th  $n$ -bit gray code has its leftmost bit fixed and it uses 1st  $(n - 1)$ -bit gray code for upper half and  $G(n - 1)$ -th  $(n - 1)$ -bit gray code for lower half.
- $[G(n - 1) + 1]$ -th  $n$ -bit gray code has its leftmost bit fixed and it uses 2nd  $(n - 1)$ -bit gray code for upper half and 1st  $(n - 1)$ -bit gray code for lower half.
- $G(n)$ -th  $n$ -bit gray code has its rightmost bit fixed and it uses  $G(n - 1)$ -th  $(n - 1)$ -bit gray code for both halves.

You have to find a  $n$ -bit gray code for given starting bit pattern and index.

### Input

The first line of the input file contains a single integer  $N$  ( $0 < N \leq 1000$ ) which denotes the number of inputs. Each of the next  $N$  lines contains a string of bits for starting bit pattern and an integer for index. Number of bits will be between 1 to 6. And the index will be valid.

### Output

Print the gray code for the given starting bit pattern and index. Put a blank line between two consecutive sets of inputs.

### Sample Input

```
3
000 1
111 5
10 2
```

### Sample Output

```
000
001
011
010
110
111
101
100

111
110
010
011
001
000
100
101

10
00
```

01

11