Problem B

John is obsessed with security. Recently he bought a new electronic lock. It is protected by a password containing n digits, where each digit is either zero or one. John decides to change the password every day. On the first day, the password is all zeroes. On each day that follows, he will select one or more digits that all have the same value and change their values (so zeroes become ones, and ones become zeroes). He must select the digits according to the following rules:

- During the first 2^n days, he must never use the same password twice.
- Each new password must come as early as possible alphabetically while not violating rule 1.

For example, if n is 2, the password on the first day is "00". The next day, he can change one or both 0's to get "01", "10" or "11". Of these possibilities, "01" comes earliest alphabetically. The next day, he can change either the 0 or the 1 to get "11" or "00". He can't choose "00" because it was already used, so he chooses "11". The next day, he can change one or both 1's to get "10", "01" or "00". He has already used "01" and "00", so he must choose "10".

Given ints n and k, return the password that comes latest alphabetically during the first k days.

The first line of the input contains t, the number of test cases, followed by $2 \cdot t$ lines. Each test case is represented by two integers n and k.

Restrictions:

- If A and B are two Strings of the same length, then A comes earlier alphabetically than B if it contains a smaller character at the first position where the Strings differ.
- $1 \leq n \leq 50$
- $1 \leq k \leq 2^n$

Example:

```
Input:
1
2
4

Output:
11
```