G. Pairwise Matching

Constraint: Time Limit: 1 seconds, Memory: 256MB

Problem description

You are given a set of n (n is always a power of 2) elements containing all integers $0,1,2,\ldots,n-1$ exactly once.

Find n/2 pairs of elements such that:

- Each element in the set is in exactly one pair.
- The sum over all pairs of the bitwise AND[i] of its elements must be exactly equal to k.

  Formally, if a_i and b_i are the elements of the i-th pair, then the following must hold:

$$\sum_{i=1}^{n/2} a_i \,\&\, b_i = k$$

  where $\&$ denotes the bitwise AND operation.

If there are many solutions, print any of them, if there is no solution, print -1 instead.

INPUT	OUTPUT
The input consists of multiple test cases. The first line contains a single integer t ($1 \le t \le 400$) — the number of test cases. Description of the test cases follows. Each test case consists of a single line with two integers n and k ($4 \le n \le 2^{16}$, n is a power of 2, $0 \le k \le n-1$). The sum of n over all test cases does not exceed 2^{16}. All test cases in each individual input will be pairwise **different**.	For each test case, if there is no solution, print a single line with the integer -1. Otherwise, print n/2 lines, the i-th of them must contain a_i and b_i, the elements in the i-th pair. If there are many solutions, print any of them. Print the pairs and the elements in the pairs in any order.

Example 1:

INPUT	OUTPUT
4 4 0 4 1 4 2 4 3	0 3 1 2 0 2 1 3 0 1 2 3 -1

Explanation:

In the first test, $(0\&3)+(1\&2)=0$.

In the second test, $(0\&2)+(1\&3)=1$.

In the third test, $(0\&1)+(2\&3)=2$.

In the fourth test, there is no solution.

---

[i] In computer programming, a **bitwise operation** operates on a bit string, a bit array or a binary numeral (considered as a bit string) at the level of its individual bits. It is a fast and simple action, basic to the higher-level arithmetic operations and directly supported by the processor. Most bitwise operations are presented as two-operand instructions where the result replaces one of the input operands.

On simple low-cost processors, typically, bitwise operations are substantially faster than division, several times faster than multiplication, and sometimes significantly faster than addition. While modern processors usually perform addition and multiplication just as fast as bitwise operations due to their longer instruction pipelines and other architectural design choices, bitwise operations do commonly use less power because of the reduced use of resources.[1]