Greatest Common Divisor

Input file:	`standard input`
Output file:	`standard output`
Time limit:	5 seconds
Memory limit:	512 megabytes

Gennady is an aspiring programmer. He is currently learning the Euclidean algorithm for computing the greatest common divisor of two positive integers.

Unfortunately, Gennady sometimes confuses the integer division operator (denoted by `div`) with the remainder operator (denoted by `mod`). As an example, 37 `div` 10 = 3 and 37 `mod` 10 = 7.

Here's Gennady's latest implementation of the Euclidean algorithm:

- *Input: two positive integers x and y.*
- *While $y > 0$:*

  Set $x = x$ `div` y, then swap x and y.
- *Output: x.*

As you can see, if Gennady used the `mod` operator instead of the `div` operator, his implementation would be correct: the algorithm above would successfully find the greatest common divisor of x and y. However, it turns out that even with this nasty bug the algorithm sometimes works correctly!

You are given an integer n. Gennady is interested in finding all input pairs (x, y) such that $1 \le x, y \le n$, the algorithm finishes, and produces the correct output. Let $(x_1, y_1), (x_2, y_2), \ldots, (x_k, y_k)$ be all such pairs in lexicographic order (for all $1 \le i < k$, either $x_i < x_{i+1}$, or $x_i = x_{i+1}$ and $y_i < y_{i+1}$).

You are also given q queries. Query i is a positive integer p_i, and you should print x_{p_i} and y_{p_i}, or report that $p_i > k$.

Input

The first line contains two integers n and q — the upper bound on the input values and the number of queries ($1 \le n, q \le 2 \cdot 10^5$).

Each of the next q lines contains a single integer p_i ($1 \le p_i \le n^2$).

Output

For each query, print two integers. These integers must either be x_{p_i} and y_{p_i}, denoting the p_i-th input pair in lexicographic order such that the algorithm finishes and produces a correct output, or `-1 -1` if there are less than p_i such pairs.

Example

standard input	standard output
10 13 1 2 3 4 5 6 7 8 9 10 11 12 13	2 2 3 3 4 2 4 4 5 5 6 6 7 7 8 8 9 3 9 9 10 4 10 10 -1 -1