🇬🇧 🇷🇺

D. Ceil Divisions

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

You have an array $a_1, a_2, \ldots, a_n$ where $a_i = i$.

In one step, you can choose two indices x and y ($x \neq y$) and set $a_x = \left\lceil \frac{a_x}{a_y} \right\rceil$ (ceiling function).

Your goal is to make array a consist of $n - 1$ ones and 1 two in no more than $n + 5$ steps. Note that you don't have to minimize the number of steps.

Input
The first line contains a single integer t ($1 \leq t \leq 1000$) — the number of test cases.

The first and only line of each test case contains the single integer n ($3 \leq n \leq 2 \cdot 10^5$) — the length of array a.

It's guaranteed that the sum of n over test cases doesn't exceed $2 \cdot 10^5$.

Output
For each test case, print the sequence of operations that will make a as $n - 1$ ones and 1 two in the following format: firstly, print one integer m ($m \leq n + 5$) — the number of operations; next print m pairs of integers x and y ($1 \leq x, y \leq n; x \neq y$) ($x$ may be greater or less than y) — the indices of the corresponding operation.

It can be proven that for the given constraints it's always possible to find a correct sequence of operations.

Example

input | Copy
```
2
3
4
```

output | Copy
```
2
3 2
3 2
3
3 4
4 2
4 2
```

Note
In the first test case, you have array $a = [1, 2, 3]$. For example, you can do the following:

1. choose 3, 2: $a_3 = \left\lceil \frac{a_3}{a_2} \right\rceil = 2$ and array $a = [1, 2, 2]$;
2. choose 3, 2: $a_3 = \left\lceil \frac{2}{2} \right\rceil = 1$ and array $a = [1, 2, 1]$.

You've got array with 2 ones and 1 two in 2 steps.
In the second test case, $a = [1, 2, 3, 4]$. For example, you can do the following:

1. choose 3, 4: $a_3 = \left\lceil \frac{3}{4} \right\rceil = 1$ and array $a = [1, 2, 1, 4]$;
2. choose 4, 2: $a_4 = \left\lceil \frac{4}{2} \right\rceil = 2$ and array $a = [1, 2, 1, 2]$;
3. choose 4, 2: $a_4 = \left\lceil \frac{2}{2} \right\rceil = 1$ and array $a = [1, 2, 1, 1]$.

→ Problem tags
brute force   constructive algorithms
math   number theory   *1700
No tag edit access

Educational Codeforces Round 101 (Rated for Div. 2)
Finished

Supported by

ITMO UNIVERSITY