Very Sparse Table

Input file:	*standard input*
Output file:	*standard output*
Time limit:	10 seconds
Memory limit:	512 mebibytes

This is an interactive problem.

You are given a directed graph G on vertices numbered 0 to n. Initially, G contains exactly n edges of the form $v \to v + 1$. Your task is to add some edges to this graph in such a way that for every two vertices v, u $(v < u)$ there exists a directed path from v to u consisting of at most three edges. There are also two additional requirements you must meet:

1. You can add an edge $a \to c$ if and only if there exists such b that edges $a \to b$ and $b \to c$ are already present in G.
2. You can add at most $6 \cdot n$ edges in total.

Interaction Protocol

The interaction starts with the interactor printing the only integer n $(0 \le n \le 2^{16})$ on a single line. After that, your program should output k $(0 \le k \le 6 \cdot n)$: the number of edges you want to add to the graph. The next k lines should contain descriptions of the added edges in the order of their addition. Each edge should be described by three integers a, b, c, meaning that you add the edge $a \to c$, and that edges $a \to b$ and $b \to c$ are already present in G.

Do not forget to flush the output buffer after this (you can use `cout << flush;` in C++), otherwise, the solution will get "`Idleness Limit Exceeded`".

Then, the interactor prints the number of queries q $(0 \le q \le 2 \cdot 10^5)$ and q queries on the next q lines. Each query is described by two integers ℓ, r $(0 \le \ell < r \le n)$ for which your program should output a line containing a path in G of length at most three that starts in vertex ℓ and finishes in vertex r. The path should be printed as a sequence of all vertices it visits (including both endpoints) separated by spaces.

To get the next query, the solution **must** print the answer to the previous query, end the line with a newline and flush the output buffer. The queries **depend** on the edges you printed.

In the example below, there are **no** actual empty lines in input and output: they are only added to visually align the inputs and the respective outputs.

Example

standard input	*standard output*
9 3 1 8 2 4 0 5	7 1 2 3 0 1 3 6 7 8 6 8 9 3 4 5 4 5 6 3 5 6 1 3 6 8 2 3 4 0 1 3 5