Problem E. Marketing

Input file:	*standard input*
Output file:	*standard output*
Time limit:	30 seconds
Memory limit:	512 mebibytes

This is an interactive problem.

Your company is producing multiple types of a product. At first, you had just one type, which you called Type 1. Then you came up with a more expensive one, and called it Type 2. Even more expensive one came about, which you called Type 3 — so far, so good.

However, then you came up with a new type that costs more than Type 2, but less than Type 3, and your customers have come to appreciate the fact that higher type numbers correspond to higher cost. So you've decided to rename the old Type 3 as Type 4, and release the new type as Type 3.

Your troubles didn't end there, as the next type you decided to produce was cheaper than all four existing ones. You've decided against using zero or negative numbers, so now you had to rename all existing types! In order to make the situation a bit more future-proof, you've renamed existing Type 1 to Type 20, existing Type 2 to Type 30, existing Type 3 to Type 40, existing Type 4 to Type 50, and added the new cheapest Type 10.

Of course even despite this clever idea you might have to rename the existing types again after a few more additions...

You've decided to automate your decisions and write a program that will assign type numbers and do the renaming for you in such a way that the total number of rename operations is not so big.

Your program will repeatedly be given where the newly added type appears in the sorted order, and needs to respond with a type number for the newly added type, as well as an optional list of rename operations it needs to do to maintain the sorted order by type numbers.

Note that the interactor will behave adaptively (some may call it *adversarily*), and the positions of newly added types given to your program will depend on the type numbers it assigned to existing types.

Interaction Protocol

The interaction consists of multiple rounds. In each round, one new type is added. Your program needs to read the integer a_i — the type number of the type such that the new type needs to come exactly after that type in the sorted order, or 0 if the new type needs to become the first type. Then your program needs to print the number b_i ($1 \le b_i \le 65535$) assigned to the newly added type, the number n_i of rename operations you want to do in this round, and n_i pairs of integers, each pair describing one rename operation: the type numbers before and after renaming, in this order. The type numbers after renaming must also be between 1 and 65535, inclusive.

Note that the rename operations will happen all at once, and before the new type is added, so you can reuse one of the type numbers being renamed for the new type, or rename one of the types to the old number of another type being renamed. Remember to flush the output after printing your decisions.

After at most 10000 rounds, your program will read the integer -1, which means that the interaction is over and your program should exit.

Before the first round, there are no types, so the very first integer you read will always be 0 or -1.

Your solution will be accepted if the type numbers after each round are distinct, within the range mentioned above, and properly sorted, and for each m the total number of rename operations after the first m rounds does not exceed $100 \cdot m$: $\sum_{i=1}^{m} n_i <= 100 \cdot m$.

Examples

standard input	standard output
0 1 2 2 0 -1	1 0 2 0 3 0 3 1 3 4 10 4 1 20 2 30 3 40 4 50
0 100 111 -1	100 0 200 1 100 111 150 0

Note

Note that the first few type numbers and renames described in the beginning of the problem statement serve as an example only, you program needs to start from an empty set of types and take all decisions by itself.

The first sample input/output corresponds to the example from the problem statement.