Task – popa

Input file:	**popa.in**
Output file:	**popa.out**
Time limit:	**1 second**
Memory limit:	**256 megabytes**

"He's an outlaw and he's famous
Andrii Popa the courageous.

Day and night he rides,
He takes his tribute from the main road
And everywhere in the country
The thief catchers are running away as fast as they can"
- *"Andrii Popa"*, Phoenix

Ghiță has a 0-indexed sequence `S` of `N` positive integer weights. Since he is the king of the Carpathians, he wants to construct a binary tree whose nodes have indices `0`, `1`, …, `N - 1` such that:

- The inorder traversal of the tree yields the nodes in increasing order of indices. The inorder traversal of a binary tree is formed out of the inorder traversal of the the subtree rooted at the root's left child (if the child exists), followed by the index of the root, followed by the inorder traversal of the subtree rooted at the root's right child (if the child exists).
- If node `x` is the parent of node `y` then S_x divides S_y.

We recall that a binary tree is a tree data structure in which each node has at most two children, which are referred to as the left child and the right child.

Unfortunately, the famous outlaw Andrii Popa has stolen the sequence S, and Ghiță can no longer access it directly. Using advanced technology (his mobile phone), he can, for any two continuous subsequence `[a, b]` and `[c, d]` of `S`, find out whether `gcd[a, b]` equals `gcd[c, d]`, where `gcd[x, y]` is the greatest common divisor of S_x , S_{x+1}, S_{x+2}, …, S_y. Unfortunately, the technology is quite expensive – if Ghiță uses it more than `Q` times, he will have to pay a large fine. Help him by using the technology at most `Q` times to build the desired tree. It is guaranteed that this is possible. Any valid solution will be accepted.

Interaction

The contestant should include the header `popa.h` (C / C++) or implement `popa.pas` (Pascal). The contestant needs to implement the following function:

```
int solve(int N, int* Left, int* Right);
function solve(N: LongInt, var Left, Right: array of LongInt): LongInt;
```

This function should return the root of the tree, and set $\texttt{Left}_i$ and $\texttt{Right}_i$ to the left and right children of the node `i` respectively. If node `i` doesn't have a left child, then $\texttt{Left}_i$ should be −1. If node `i` doesn't have a right child, then $\texttt{Right}_i$ should be −1. `Left` and `Right` point to two arrays of sizes exactly N that have already been allocated.

The function `solve` may be called **at most 5 times** over the course of a single run. We suggest being careful with global variables.

The contestant may use the following function:

```
int query(int a, int b, int c, int d);
function query(a, b, c, d: LongInt): LongInt;
```

This function will return 1 if and only if gcd[a, b] is equal to gcd[c, d], where $0 \le a \le b < N$ and $0 \le c \le d < N$, and 0 otherwise.

Scoring

Subtask 1 (37 points)

- N = 100
- Q = 10.000

Subtask 2 (24 points)

- N = 1.000
- Q = 20.000

Subtask 3 (39 points)

- N = 1.000
- Q = 2.000

Example

An example of the interaction for S = [12, 4, 16, 2, 2, 20] is:

Calls of **solve**	Calls of **query**	After the call of **solve**
solve(6, Left, Right)		
	query(0, 1, 3, 5) returns **0**	
	query(4, 5, 1, 3) returns **1**	
		solve returns **3** **Left** points to **[-1, 0, -1, 1, -1, -1]** **Right** points to **[-1, 2, -1, 4, 5, -1]**

Notes

An example of a tree of the kind that Ghiță wants for the previous example is: