Fredo and Large Numbers

Max. Marks: 100

Fredo is pretty good at dealing large numbers. So, once his friend Zeus gave him an array of **N** numbers , followed by **Q** queries which he has to answer. In each query , he defines the type of the query and the number **f** for which Fredo has to answer. Each query is of the following two types:
Type 0: For this query, Fredo has to answer the first number in the array (starting from index 0) such that its frequency is atleast equal to **f**.
Type 1: For this query, Fredo has to answer the first number in the array such that frequecy is exactly equal to **f**.
Now, Fredo answers all his queries but now Zeus imagines how he should verify them . So, he asks you to write a code for the same.
Note: If there is no number which is the answer to the query, output 0.
Use fast I/O.

Input :
The first line of the input contains **N** , the size of the array
The next line contains **N** space separated integers.
The next line contains **Q**, denoting the number of queries.
Then follow **Q** lines, each line having two integers type and f, denoting the type of query and the frequency for which you have to answer the query.

Output:
You have to print the answer for each query in a separate line.

Input Constraints:
$1 \leq N \leq 10^6$
$1 \leq A[i] \leq 10^{18}$
$1 \leq Q \leq 10^6$
$0 \leq type \leq 1$
$1 \leq f \leq 10^{18}$

SAMPLE INPUT

```
6
1 2 2 1 2 3
5
0 1
0 2
1 2
1 3
0 3
```

SAMPLE OUTPUT

```
1
1
1
2
2
```

Explanation

Query 1: 1 is the first number from left with frequency atleast 1.
Query 2: 1 is the first number from left with frequency atleast 2.
Query 3: 1 is the first number from left with frequency exactly 2.
Query 4: 2 is the first number from left with frequency exactly 3.
Query 5: 2 is the first number from left with frequency atleast 3.

Time Limit:	1.0 sec(s) for each input file.
Memory Limit:	256 MB
Source Limit:	1024 KB
Marking Scheme:	Marks are awarded if any testcase passes.
Allowed Languages:	C, CPP, CLOJURE, CSHARP, D, ERLANG, FSHARP, GO, GROOVY, HASKELL, JAVA, JAVA8, JAVASCRIPT, JAVASCRIPT_NODE, LISP, LISP_SBCL, LUA, OBJECTIVEC, OCAML, OCTAVE, PASCAL, PERL, PHP, PYTHON, PYTHON3, R, RACKET, RUBY, RUST, SCALA, SWIFT, VB

CODE EDITOR

😐 Our compiler wanted to be here!

But the mobile is too cramped for it to load. It says it would be more comfortable on the web.