DMOPC '21 Contest 2 P6 - Strange Function

Time limit: 2.0s **Memory limit:** 256M

You are given a permutation $p_1, p_2, \ldots, p_N$ of $1, 2, \ldots, N$. Consider the following function:

```
void f(int l,int r) {
    if (l >= r) return;
    int i = min_element(p+l,p+r+1)-p; // position of minimum element in [l,r]
    rotate(p+l,p+i,p+i+1); // cyclically shift [l,i] to the right by 1, so that the minimum element gets moved to position l
    f(i+1,r); // continue on [i+1,r]
}
```

For example, calling $f(1, N)$ on the permutation $[4, 2, 1, 3, 6, 5]$ gives a result of $[1, 4, 2, 3, 5, 6]$. Note that p remains a permutation of $1, 2, \ldots, N$ when $f(1, N)$ is called on it several times.

Process Q queries of the form:

- `1`: Call $f(1, N)$ once.
- `2 i`: Output the current value of p_i.
- `3 i`: Output the unique index j such that $p_j = i$.

Constraints

$1 \leq N, Q \leq 3 \times 10^5$

$p_1, p_2, \ldots, p_N$ is a permutation of $1, 2, \ldots, N$.

For all type `2` and type `3` queries, $1 \leq i \leq N$.

Subtask 1 [10%]

$1 \leq N, Q \leq 5 \times 10^3$

Subtask 2 [10%]

$1 \leq N \leq 5 \times 10^3$

Subtask 3 [20%]

All type `1` queries appear before all type `2` and type `3` queries.

Subtask 4 [25%]

There are only type `1` and type `2` queries.

Subtask 5 [25%]

There are only type `1` and type `3` queries.

Subtask 6 [10%]

No additional constraints.

Input Specification

The first line contains two space-separated integers: N and Q.

The second line contains N space-separated integers: $p_1, p_2, \ldots, p_N$.

The next Q lines each describe a query.

Output Specification

For each type `2` or type `3` query, output the answer on a new line.

Sample Input

```
6 16
4 2 1 3 6 5
3 1
3 2
3 3
3 4
3 5
3 6
1
2 1
2 2
2 3
2 4
2 5
2 6
1
2 3
3 3
```

Sample Output

```
3
2
4
1
6
5
1
4
2
3
5
6
4
4
```

Explanation

Initially, p is $[4, 2, 1, 3, 6, 5]$. We have

- 1 appears at index 3, so the answer to the first query is 3.
- 2 appears at index 2, so the answer to the second query is 2.
- 3 appears at index 4, so the answer to the third query is 4.
- 4 appears at index 1, so the answer to the fourth query is 1.
- 5 appears at index 6, so the answer to the fifth query is 6.
- 6 appears at index 5, so the answer to the sixth query is 5.

After calling $f(1, N)$ once, p is $[1, 4, 2, 3, 5, 6]$.

After calling $f(1, N)$ a second time, p is $[1, 2, 4, 3, 5, 6]$.