Choreography

The Hungarian National Dance Ensemble is practicing a new choreography. There are N dancers in the ensemble, numbered from 0 to $N-1$, where N is an *even* number.

At the practice sessions, the choreographer first lines up the dancers. The positions in the line are numbered from 0 to $N-1$, and position i is occupied by dancer $P[i]$ at the start.

After the dancers line up, the choreographer asks them to perform a sequence of **move**s. Each move must be completed before they begin performing the next one. The dancers are practicing 4 types of moves:

1. Every dancer moves K positions to the right *cyclically*, where $0 \leq K < N$. That is,
   - for each i from 0 to $N-K-1$, inclusive, the dancer currently standing at position i moves to position $i+K$, and
   - for each i from $N-K$ to $N-1$, inclusive, the dancer currently standing at position i moves to position $i+K-N$.
2. Every dancer moves K positions to the left *cyclically*, where $0 \leq K < N$. That is,
   - for each i from K to $N-1$, inclusive, the dancer currently standing at position i moves to position $i-K$, and
   - for each i from 0 to $K-1$, inclusive, the dancer currently standing at position i moves to position $i-K+N$.
3. For each i from 0 to $\frac{N}{2}-1$, inclusive, dancers at positions $2i$ and $2i+1$ swap places.
4. Suppose that position i ($0 \leq i < N$) is occupied by dancer r_i before performing this move. For each j from 0 to $N-1$, inclusive, *dancer* j moves to position r_j.

Note that the positions of the dancers are distinct at the end of each move.

Before the next session, the choreographer plans a sequence of M moves for the ensemble to practice. He adds the moves to the sequence one by one. Sometimes, before adding the next move, he wonders which positions certain dancers will occupy right after performing every move that has been added to the sequence.

Your task is to simulate the dance moves planned by the choreographer and answer his questions about the positions of the dancers.

Implementation Details

You should implement the following procedures:

```
void init(int N, int[] P)
```

- N: the number of dancers.
- P: array of length N describing the initial order of the dancers.
- This procedure is called exactly once, before any other function calls.

```
void move_right(int K)
```

- K: the number of positions each dancer moves to the right.
- This procedure adds a move of type 1 to the sequence of moves.

```
void move_left(int K)
```

- K: the number of positions each dancer moves to the left.
- This procedure adds a move of type 2 to the sequence of moves.

```
void swap_places()
```

- This procedure adds a move of type 3 to the sequence of moves.

```
void move_around()
```

- This procedure adds a move of type 4 to the sequence of moves.

Procedures `move_right`, `move_left`, `swap_places`, and `move_around` are called for a total of M times.

```
int get_position(int D)
```

- D: an integer representing a dancer.
- This procedure should return the position of dancer D after performing every move added to the sequence of moves before this call.
- This procedure is called Q times.

Example

Consider the following sequence of calls:

```
init(6, [5, 1, 4, 2, 0, 3])
```

There are 6 dancers, and the initial order of the dancers is given by the array $[5, 1, 4, 2, 0, 3]$, that is, position 0 is occupied by dancer 5, position 1 is occupied by dancer 1, position 2 is occupied by

dancer 4, and so on.

```
move_left(2)
```

Each dancer moves two places to the left cyclically. After the move, the order of the dancers becomes $[4, 2, 0, 3, 5, 1]$.

```
get_position(0)
```

This call asks the position of dancer 0 after performing the first move. Dancer 0 is standing at position 2. Therefore, the procedure should return 2.

```
swap_places()
```

After this move, the order of the dancers becomes $[2, 4, 3, 0, 1, 5]$.

```
move_around()
```

The new position of each dancer is determined as follows.

- Dancer 0: position 0 is occupied by dancer 2, so dancer 0 moves to position 2.
- Dancer 1: position 1 is occupied by dancer 4, so dancer 1 stays at position 4.
- Dancer 2: position 2 is occupied by dancer 3, so dancer 2 moves to position 3.
- Dancer 3: position 3 is occupied by dancer 0, so dancer 3 moves to position 0.
- Dancer 4: position 4 is occupied by dancer 1, so dancer 4 stays at position 1.
- Dancer 5: position 5 is occupied by dancer 5, so dancer 5 stays at position 5.

After this move, the order of the dancers becomes $[3, 4, 0, 2, 1, 5]$.

```
get_position(3)
```

Dancer 3 is at position 0 after performing every move so far. The procedure should return 0.

Constraints

- $1 \leq N \leq 100\,000$
- $0 \leq P[i] < N$ (for each i such that $0 \leq i < N$)
- $P[i] \neq P[j]$ (for each i and j such that $0 \leq i < j < N$)
- $0 \leq M \leq 100\,000$
- $1 \leq Q \leq 200\,000$
- $0 \leq K < N$
- $0 \leq D < N$

Subtasks

1. (7 points) Only moves of types 1 and 2 are performed.
2. (12 points) $N \cdot (Q + M) \leq 1\,000\,000$
3. (10 points) Only moves of type 4 are performed.
4. (14 points) Only moves of types 3 and 4 are performed.
5. (28 points) Only moves of types 1, 2, and 4 are performed.
6. (29 points) No additional constraints.

Sample Grader

The sample grader reads the input in the following format:

- line 1: $N\ M\ Q$
- line 2: $P[0]\ P[1]\ \ldots\ P[N-1]$
- line $3 + l$ ($0 \leq l < M + Q$):
  - To perform a move of type 1: $1\ K$
  - To perform a move of type 2: $2\ K$
  - To perform a move of type 3: 3
  - To perform a move of type 4: 4
  - To ask the position of a dancer: $5\ D$

The sample grader prints your answers in the following format:

- line $1 + q$ ($0 \leq q < Q$): the values returned by `get_position`