드문 곤충

블랑콘씨의 집 주위를 돌아다니는 N 마리의 곤충이 있고, 각각 0에서 $N-1$로 번호가 매겨져 있다. 각 곤충마다 자신이 속하는 **종류**가 있고, 0 이상 10^9 이하인 정수로 표현된다. 복수의 곤충이 같은 종류에 속할 수 있다.

곤충들을 종류에 따라 그룹으로 모아보기로 하자. **가장 자주 나오는** 곤충 종류 그룹의 크기는 가장 많은 수의 곤충이 속하는 그룹의 곤충 수이다. 비슷하게, **가장 드문** 곤충 종류 그룹의 크기는 가장 적은 수의 곤충이 속하는 그룹의 곤충 수이다.

예를 들어, 11 마리의 곤충이 있고, 종류가 차례로 $[5, 7, 9, 11, 11, 5, 0, 11, 9, 100, 9]$라고 하자. 이 경우, **가장 자주 나오는** 곤충 종류 그룹의 크기는 3이다. 종류 9인 곤충의 그룹과 종류 11인 곤충의 그룹이 가장 자주 나오는 곤충 종류 그룹이며, 각각 3 마리의 곤충으로 이루어져 있다. **가장 드문** 곤충 종류 그룹의 크기는 1이다. 종류 7, 종류 0, 종류 100인 곤충의 그룹에는 각각 1 마리의 곤충이 속한다.

블랑콘씨는 곤충의 종류를 알지 못한다. 곤충의 종류에 대한 정보를 알려주는 단추가 하나 달린 기계가 있다. 처음, 이 기계는 비어 있다. 기계를 이용하기 위해, 다음 세 가지 종류의 연산을 사용할 수 있다.

1. 곤충 한 마리를 기계 안에 넣는다.
2. 곤충 한 마리를 기계에서 빼낸다.
3. 기계의 단추를 누른다.

각각의 연산은 최대 $40\ 000$ 번 수행될 수 있다.

단추가 눌려질 때마다, 이 기계는 현재 기계에 들어 있는 곤충들만 이용해서, 이 중 **가장 자주 나오는** 곤충 종류 그룹의 크기를 알려준다.

당신이 할 일은 이 기계를 사용해서 블랑콘씨 집 주위의 N 마리 곤충들 중 **가장 드문** 곤충 종류 그룹의 크기를 구하는 것이다. 덤으로, 어떤 서브태스크에서는, 특정한 명령이 실행되는 횟수에 따라 여러분의 점수가 결정된다 (자세한 내용은 Subtasks 참고).

Implementation Details

다음 함수를 구현해야 한다.

```
int min_cardinality(int N)
```

- N: 곤충의 수
- 이 함수는 블랑콘씨 집 주위에 있는 모든 N 마리 곤충들 중 **가장 드문** 곤충 종류 그룹의 크기를 리턴해야 한다.

- 이 함수는 정확히 한 번 호출된다.

위 함수는 다음 함수들을 호출할 수 있다.

```
void move_inside(int i)
```

- i: 기계에 집어넣을 곤충의 인덱스. i 값은 0 이상 $N-1$ 이하여야 한다.
- 만약 이 곤충이 이미 기계 안에 있다면, 이 함수 호출은 기계 안에 있는 곤충의 집합을 바꾸지 않는다. 하지만 이 호출도 별도의 함수 호출 한 번으로 계산된다.
- 이 함수는 최대 $40\ 000$ 번 호출될 수 있다.

```
void move_outside(int i)
```

- i: 기계에서 뺄 곤충의 인덱스. i 값은 0 이상 $N-1$ 이하여야 한다.
- 만약 이 곤충이 이미 기계 밖에 있다면, 이 함수 호출은 기계 안에 있는 곤충의 집합을 바꾸지 않는다. 하지만 이 호출도 별도의 함수 호출 한 번으로 계산된다.
- 이 함수는 최대 $40\ 000$ 번 호출될 수 있다.

```
int press_button()
```

- 이 함수는 기계 안에 있는 곤충들 중에서, **가장 자주 나오는** 곤충 종류 그룹의 크기를 알려준다. 전체 곤충이 아니고 기계 안에 있는 곤충만 고려한다는데 유의하라.
- 이 함수는 최대 $40\ 000$ 번 호출될 수 있다.
- 그레이더는 **적응적(adaptive)이지 않다.** 즉, 곤충 N 마리의 종류는 `min_cardinality` 함수를 호출하기 전에 이미 결정되어 있다.

Example

6 마리 곤충이 있고, 종류가 차례로 $[5, 8, 9, 5, 9, 9]$인 시나리오를 생각해보자. `min_cardinality` 함수는 다음과 같이 호출된다.

```
min_cardinality(6)
```

이 함수는 `move_inside`, `move_outside`, `press_button`를 다음과 같이 호출한다.

호출하는 함수	리턴 값	기계 안에 있는 곤충	기계 안의 곤충들 종류
		$\{\}$	$[]$
`move_inside(0)`		$\{0\}$	$[5]$
`press_button()`	1	$\{0\}$	$[5]$
`move_inside(1)`		$\{0, 1\}$	$[5, 8]$
`press_button()`	1	$\{0, 1\}$	$[5, 8]$
`move_inside(3)`		$\{0, 1, 3\}$	$[5, 8, 5]$
`press_button()`	2	$\{0, 1, 3\}$	$[5, 8, 5]$
`move_inside(2)`		$\{0, 1, 2, 3\}$	$[5, 8, 9, 5]$
`move_inside(4)`		$\{0, 1, 2, 3, 4\}$	$[5, 8, 9, 5, 9]$
`move_inside(5)`		$\{0, 1, 2, 3, 4, 5\}$	$[5, 8, 9, 5, 9, 9]$
`press_button()`	3	$\{0, 1, 2, 3, 4, 5\}$	$[5, 8, 9, 5, 9, 9]$
`move_inside(5)`		$\{0, 1, 2, 3, 4, 5\}$	$[5, 8, 9, 5, 9, 9]$
`press_button()`	3	$\{0, 1, 2, 3, 4, 5\}$	$[5, 8, 9, 5, 9, 9]$
`move_outside(5)`		$\{0, 1, 2, 3, 4\}$	$[5, 8, 9, 5, 9]$
`press_button()`	2	$\{0, 1, 2, 3, 4\}$	$[5, 8, 9, 5, 9]$

이 시점에서, 가장 드문 곤충 종류 그룹의 크기가 1 임을 아는데 충분한 정보를 얻게 된다. 따라서, `min_cardinality` 함수의 리턴값은 1이어야 한다.

이 예제에서, `move_inside` 함수는 7 번, `move_outside` 함수는 1 번, `press_button` 함수는 6 번 호출되었다.

Constraints

- $2 \leq N \leq 2000$

Subtasks

1. (10 points) $N \leq 200$
2. (15 points) $N \leq 1000$
3. (75 points) 추가적인 제약 조건이 없다.

어떤 테스트 케이스에서든, 함수 `move_inside`, `move_outside`, `press_button`를 호출할 때 Implementation Details에서 설명한 제약 조건을 만족하지 않았다거나, `min_cardinality` 함수의 출력이 틀렸다면, 해당 서브태스크에서 여러분의 점수는 0점이다.

q가 `move_inside` 함수 호출 횟수, `move_outside` 함수 호출 횟수, `press_button` 함수 호출 횟수 세 값 중 **최대값**이라고 하자.

서브태스크 3에서는 부분 점수가 있다. m이 이 서브태스크의 모든 테스트케이스에서 $\frac{q}{N}$의 최대값이라고 하자. 이 서브태스크에서 얻을 수 있는 점수는 다음 표에 따라 계산된다.

조건	점수
$20 < m$	0 (CMS에서 "`Output isn't correct`"로 출력)
$6 < m \leq 20$	$\frac{225}{m-2}$
$3 < m \leq 6$	$81 - \frac{2}{3}m^2$
$m \leq 3$	75

Sample Grader

T가 길이 N인 정수 배열이고 $T[i]$가 곤충 i의 종류라고 하자.

샘플 그레이더는 다음 양식으로 입력을 읽는다.

- line 1: N
- line 2: $T[0]$ $T[1]$ $\ldots$ $T[N-1]$

만약 샘플 그레이더가 프로토콜이 위반된 것을 발견하면, 샘플 그레이더는 `Protocol Violation: <MSG>`를 출력하는데,`<MSG>`는 다음 중 하나이다.

- `invalid parameter`: `move_inside` 또는 `move_outside`를 호출할 때, i 값이 0 이상 $N-1$ 이하가 아니다.
- `too many calls`: `move_inside`, `move_outside`, `press_button` 중 **어떤 한** 함수가 $40\,000$ 번 초과되어 호출되었다.

그 외의 경우에는, 샘플 그레이더는 다음 양식에 따라 출력한다.

- line 1: `min_cardinality`의 리턴값
- line 2: q