1-3. List of Unique Integers

There is a hidden integer sequence $A[0], A[1], \cdots, A[N-1]$. You have to write a program that finds all indices i $(0 \le i \le N-1)$ where $A[i]$ is unique among $A[0], \cdots, A[N-1]$.

What you can do is to ask the grader the following question multiple times: what is the number of unique integers among $A[L..R]$?

Implementation details

You should implement the following function:

```
int[] PickUnique(int N)
```

- `N`: the length of the hidden sequence.
- This function is called exactly once for each test case.
- This function should return an array U of length N. For each i $(0 \le i \le N-1)$, $U[i]$ should be 1 if $A[i]$ is unique among the hidden sequence A, and 0 otherwise.

The function `PickUnique` can call the following grader function:

```
int UniqueCount(int L, int R)
```

- L, R: the leftmost and rightmost indices of the subsegment. $0 \le L \le R \le N-1$ should hold.
- This function returns the number of unique integers among $A[L]$, $A[L+1]$, $\cdots$, $A[R-1]$, $A[R]$. In other words, it returns the number of indices j $(L \le j \le R)$ where $A[j]$ appears exactly once in $A[L..R]$.
- This function can be called at most $40\,000$ times.

If some of the above conditions are not satisfied, your program is judged as `Wrong Answer`. Otherwise, your program is judged as `Accepted` and your score is calculated by the number of calls to `UniqueCount` (see Subtasks).

Example

Assume that the sequence A is $[1, 2, 3, 1, 2]$. The grader makes the following function call:

```
PickUnique(5)
```

Let us consider the following calls to the function `UniqueCount`:

- `UniqueCount(0, 4)`: $A[0..4] = [1, 2, 3, 1, 2]$. 1 and 2 appears twice, while 3 appears exactly once. Thus, the function returns 1.
- `UniqueCount(1, 3)`: $A[1..3] = [2, 3, 1]$. Each number appears exactly once, so the function returns 3.

The answer is $[0, 0, 1, 0, 0]$.

Constraints

- $2 \leq N \leq 200$
- $1 \leq A[i] \leq 200$ (for all $0 \leq i \leq N - 1$)

In this problem, the grader is NOT adaptive. This means N and A are fixed at the beginning of the running of the grader and they do not depend on the queries asked by your solution.

Subtasks

1. (100 points) No additional constraints.

Assume your program is judged as `Accepted`, and makes C calls to `UniqueCount`. Then your score P for the test case is calculated as follows:

- If $C \leq 400$, $P = 100$.
- If $401 \leq C \leq 40\,000$, $P = 35$.
- If $C > 40\,000$, $P = 0$.

Your score is the minimum of the scores for the test cases.

Sample grader

You can download the sample grader package on the same page you downloaded the problem statement. (scroll down if you don't see the attachment)

If you use IDEs like Visual Studio, Eclipse or Code::Blocks, then import `unique.cpp`, `unique.h` and `grader.cpp` into one project and you will be able to compile all these files at once.

If you want to compile by yourself, refer to the compilation commands in the statement page.

You should submit only `unique.cpp`.

Input format

- line 1: N
- line 2: $A[0]\ A[1]\ \cdots A[N-1]$

Output format

If your program is judged as `Accepted`, the sample grader prints `Correct` in the first line and q in the second line, with q the number of calls to `UniqueCount`.

If your program is judged as `Wrong Answer`, the sample grader prints the error message in the first line.