Underrated

There are N contestants participating in Stelele Informaticii, numbered $0, 1, \ldots, N-1$.

After the contest, the contestants are ranked according to a secret permutation P of $0, 1, \ldots, N-1$, where $P[i]$ is the position of contestant i in the final ranking. Position 0 corresponds to first place. Álec Sdima knows the final ranking, but you do not.

You may propose a ranking Q, which must also be a permutation of $0, 1, \ldots, N-1$, where $Q[i]$ is the position you assign to contestant i. The underrating of contestant i in your ranking is defined as $Q[i] - P[i]$.

After seeing your proposed ranking, Álec tells you the maximum underrating among all contestants, that is,

$$\max_{0 \le i < N} (Q[i] - P[i]).$$

Your task is to determine the secret ranking P while making as few queries as possible.

Implementation details

You should implement the following procedure.

```
std::vector<int> find_ranking(int N)
```

- N: the number of contestants.
- This procedure should return the secret permutation P.
- This procedure is called exactly once for each test case.

The above procedure can make calls to the following procedure:

```
int ask(std::vector<int> Q)
```

- Q must be a permutation of $0, 1, \ldots, N-1$.
- $Q[i]$ represents the position assigned to contestant i.
- The procedure returns

$$\max_{0 \le i < N} (Q[i] - P[i]).$$

- The procedure can be called at most $20\,000$ times in each test case.

The behavior of the grader is **adaptive**. This means that the grader does not have a fixed permutation P. Instead, the values returned by `ask` may depend on the previous calls to `ask`.

It is guaranteed that the grader answers in such a way that after each call there is at least one permutation P consistent with all the answers given so far.

Constraints

- $1 \leq N \leq 1000$
- P is a permutation of $0, 1, \ldots, N-1$.

Subtasks and Scoring

If, in any of the test cases, a call to the procedure `ask` does not conform to the rules mentioned above, or the permutation returned by `find_ranking` is incorrect, the score of your solution for that test case will be 0.

Otherwise, let Q be the maximum number of calls to the procedure `ask` among all test cases. Also let X be equal to

$$X = \min\left(1, \frac{2}{1 + \left(\frac{Q}{1997}\right)^{3/2}}\right).$$

Then, the score is calculated as follows:

Subtask	Score	Additional Constraints
1	10	$N \leq 100$
2	$90 \cdot X$	$N = 1000$

In particular, to obtain 100 points, your solution must use at most $1\,997$ calls to `ask` in every test case. The score of each subtask is rounded to two decimal places.

Example

Suppose $N = 4$ and the secret ranking is $P = [2, 0, 3, 1]$. This means that contestant 1 is ranked first, contestant 3 is ranked second, contestant 0 is ranked third, and contestant 2 is ranked fourth.

Consider the following call:

```
ask({0, 1, 2, 3})
```

For this ranking, the values $Q[i] - P[i]$ are $[-2, 1, -1, 2]$. Therefore, the maximum underrating is 2, so the procedure returns 2.

Now consider the call:

```
ask({3, 0, 2, 1})
```

The values $Q[i] - P[i]$ are $[1, 0, -1, 0]$. Therefore, the procedure returns 1.

Finally, if the proposed ranking is equal to the secret ranking,

```
ask({2, 0, 3, 1})
```

all values $Q[i] - P[i]$ are 0, so the procedure returns 0.

Sample Grader

The sample grader is **not adaptive**. The secret permutation P is fixed before `find_ranking` is called.

The sample grader reads the input in the following format:

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

For each call to `ask(Q)`, the sample grader returns

$$\max_{0 \le i < N} (Q[i] - P[i]).$$

The sample grader prints the output in the following format:

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

Here, C is the number of calls made to `ask`, and R is the permutation returned by `find_ranking`.