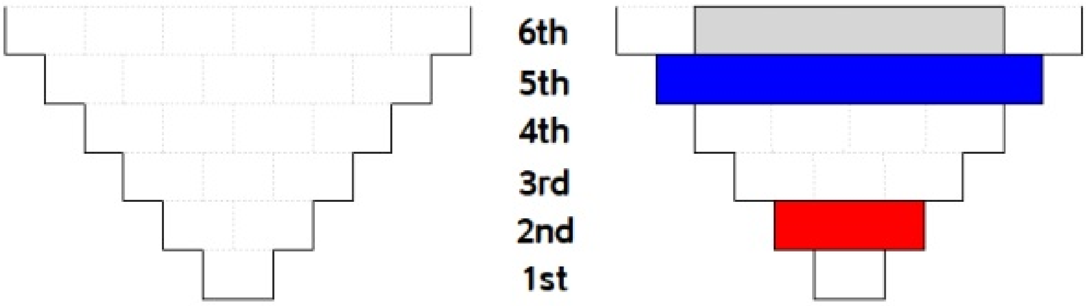
2-1. On the Grid

Jeongguk has a triangular grid which looks like the picture below. The bottom layer is numbered by 1, and the layer number increases by 1 as the layer goes up. The number of layers of the grid is sufficiently large so that we could push sticks as much as we want. Usually, N sticks of length $1, 2, \cdots, N$ are stacked in the grid (from bottom to top). But today is Jeongguk's birthday! To celebrate his birthday, N friends numbered from 0 through $N-1$ took one stick each and they are all playing with their stick. Friend i ($0 \leq i \leq N-1$) took a stick of length $L[i]$.



Jeongguk devised a new game to play with his friends using his grid. Each round of the game consists of these steps:

- First, he touches all of his friends once in some order. Right after Jeongguk touches a friend, he or she is going to put her or his stick into the grid.
  - The center of each pushed stick should always lie on the symmetric line of the grid.
  - Each stick is pushed downwards until it coincides with an already pushed stick or the border of the grid.
- Now, all N sticks are inside the grid. Jeongguk can see the highest stick of the grid and check the number of the layer that contains the highest stick.
- N friends retake their stick, and the grid becomes empty.

By repeating several rounds of the game, Jeongguk will be able to know what sticks each of the N friends took, i.e., the sequence L. Help Jeongguk to achieve this goal by touching friends as less as possible.

Implementation details

You should implement the following function. It will be called by the grader once for each test case.

```
int[] SortDisks(int N)
```

- N: number of friends.
- This function should return an array X of length N. For each $0 \le i \le N-1$, the value of $X[i]$ should be equal to $L[i]$, the length of the stick taken by friend i.

The function `SortDisks` can call the following function:

```
int PutDisks(int[] P)
```

- P: an array of length N. P should be a permutation of $0, 1, \cdots, N-1$. For each i ($0 \le i \le N-1$), the $(i+1)$-th friend Jeongguk will touch is friend $P[i]$.
- This function returns the number of the layer that contains the highest stick.
- This function can be called at most $2\,000$ times.

If some of the above conditions are not satisfied, your program is judged as `Wrong Answer`. Otherwise, your program is judged as `Accepted`.

Constraints

- $1 \le N \le 1\,000$
- $1 \le L[i] \le N$ (for all $0 \le i \le N-1$)
- $L[i] \ne L[j]$ (for all $0 \le i < j \le N-1$)

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

Subtasks

1. (12 points) $N \le 50$.
2. (31 points) $N \le 250$.
3. (57 points) No additional constraints.

Example

Assume the sequence L is $[1, 4, 2, 3]$. The grader makes the following function call:

```
SortDisks(4)
```

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

- `PutDisks([0, 2, 3, 1])`: The length of friend 0, friend 2, friend 3, and friend 1's

stick is 1, 2, 3, 4, respectively. Their sticks are put on the 1st, 2nd, 3rd, and 4th layer, respectively. Thus, the function returns 4.
- `PutDisks([2, 1, 0, 3])`: The length of friend 2, friend 1, friend 0, and friend 3's stick is 2, 4, 1, 3. Their sticks are put on the 2nd, 4th, 5th, and 6th layer, respectively. Thus, the function returns 6.

The answer is $[1, 4, 2, 3]$.

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 `grid.cpp`, `grid.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 `grid.cpp`.

Input format

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

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 `PutDisks`.

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