E. Mike and code of a permutation

time limit per test: 4 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

Mike has discovered a new way to encode permutations. If he has a permutation $P = [p_1, p_2, ..., p_n]$, he will encode it in the following way:

Denote by $A = [a_1, a_2, ..., a_n]$ a sequence of length n which will represent the code of the permutation. For each i from 1 to n sequentially, he will choose the smallest unmarked j $(1 \le j \le n)$ such that $p_i < p_j$ and will assign to a_i the number j (in other words he performs $a_i = j$) and will mark j. If there is no such j, he'll assign to a_i the number -1 (he performs $a_i = -1$).

Mike forgot his original permutation but he remembers its code. Your task is simple: find **any** permutation such that its code is the same as the code of Mike's original permutation.

You may assume that there will always be at least one valid permutation.

Input

The first line contains single integer n $(1 \le n \le 500\,000)$ — length of permutation.

The second line contains n space-separated integers $a_1, a_2, ..., a_n$ $(1 \le a_i \le n$ or $a_i = -1)$ — the code of Mike's permutation.

You may assume that all positive values from A are different.

Output

In first and only line print n numbers $p_1, p_2, ..., p_n$ $(1 \le p_i \le n)$ — a permutation P which has the same code as the given one. Note that numbers in permutation are distinct.

Examples

input

```
6
2 -1 1 5 -1 4
```

output

```
2 6 1 4 5 3
```

input

```
8
2 -1 4 -1 6 -1 8 -1
```

output

```
1 8 2 7 3 6 4 5
```

Note

For the permutation from the first example:

$i = 1$, the smallest j is 2 because $p_2 = 6 > p_1 = 2$.

$i = 2$, there is no j because $p_2 = 6$ is the greatest element in the permutation.

$i = 3$, the smallest j is 1 because $p_1 = 2 > p_3 = 1$.

$i = 4$, the smallest j is 5 (2 was already marked) because $p_5 = 5 > p_4 = 4$.

$i = 5$, there is no j because 2 is already marked.

$i = 6$, the smallest j is 4 because $p_4 = 4 > p_6 = 3$.