

Problem C

Sodium Sort

Time limit: 2s

Memory limit: 1GB

Sodium has implemented his own version of the Quicksort algorithm, called Sodium Sort, to sort an array a of length n . To choose the pivot for the partitioning step, he uses a pre-generated sequence of integers R .

Sodium Sort is implemented with the following pseudo-code:

```
int it = 1;
long long tot_length = 0;

void sodium_sort(int l, int r) {
    if (l > r) return;
    tot_length += r - l + 1;
    int pivot_index = l + (R[it] % (r - l + 1));
    it++;
    int p = partition(l, r, pivot_index);
    sodium_sort(l, p - 1);
    sodium_sort(p + 1, r);
}
```

It can be seen that the global variable `tot_length` acts as a cost measure representing the total length of all subarrays processed during the sort. Furthermore, the global counter `it` is incremented exactly n times when sorting an array of size n .

When `partition(l, r, pivot_index)` is called on a subarray, it uses the element at index `pivot_index` as the pivot value x and rearranges the subarray $a[l \dots r]$ such that:

- All elements less than x are moved to the left of the pivot.
- All elements greater than x are moved to the right of the pivot.
- Elements placed in the left and right subarrays retain their relative order from immediately before the current call. For example: When calling `partition(1, 5, 5)` on $[1, 5, 2, 4, 3]$ (where the pivot value is $x = 3$), the final array becomes $[1, 2, 3, 5, 4]$.
- It returns the new index p where the pivot x is finally placed.

Sodium will initially call `sodium_sort(1, n)` to sort his array. Sodium Sort always sorts the array correctly, but its running time depends on the initial order of the elements. Sodium's friend Sulfur wants to hack Sodium Sort by finding an input that makes it as slow as possible.

Your task is to help Sulfur. Given the length of the array n and the sequence R , construct a permutation of the integers from 1 to n for the array a such that the final value of the global counter `tot_length` is maximized.

Input

- The first line contains a single integer n ($1 \leq n \leq 2 \times 10^5$) — the size of the array to be sorted.
- The second line contains n integers $R_1, R_2, \dots, R_n$ ($0 \leq R_i \leq 10^9$) — the elements of the pre-generated sequence used for Sodium's pivot selection.

Output

Output a single line containing n integers, representing a permutation of $1, 2, \dots, n$ that maximizes the final value of the counter `tot_length` after the array is completely sorted. (A permutation of length n is an array containing each integer from 1 to n exactly once). If there are multiple valid permutations, you may output any of them.

Sample Input	Sample Output
3 5 2 9	2 3 1