A. Build A Heap

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Use manual heap implementation, do not use sets/priority queues from the standard library. Use classic linear-time algorithm for building a heap discussed on lecture.

You are given n integers. Build a heap and output a sequence of swaps that you used to build the heap.

Input

The first line contains a single integer n ($1 \le n \le 10^5$).

The second line contains n integers $a_0, a_1, \ldots, a_{n-1}$ ($-10^9 \le a_i \le 10^9$) — the elements you need to build the heap from.

Output

In the first line print a single integer k ($0 \le k \le 3 \cdot n$) — the number of swaps you used.

In each of the next k lines output two integers i and j ($0 \le i, j < n$) — the indices of elements that you swap.

In the last line print n integers — the built heap. The heap conditions $a_i \ge a_{2i+1}$ and $a_i \ge a_{2i+2}$ should be satisfied for all suitable i.

Examples

input Copy

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

output Copy

```
3
1 4
0 1
1 3
5 4 2 1 3
```

input Copy

```
10
-1 -1 -1 -1 -1 1 1 1 1 1
```

output Copy

```
7
4 9
3 7
2 5
1 3
3 8
0 1
1 3
1 1 1 -1 1 -1 1 -1 -1 -1
```

B. Heapsort

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Use manual heap implementation, do not use sets/priority queues from the standard library.

You are given n integers. Sort them using heapsort and output the number of swaps you made in `heapify`.

Use `heapify` to build a heap and then each time after you swap the head with the last element.

In `heapify`, swap only if the current element is strictly less than the greater son. Always swap with the greater son, if they are equal, swap with the first son.

Input

The first line contains a single integer n ($1 \le n \le 10^5$).

The second line contains n integers $a_0, a_1, \ldots, a_{n-1}$ ($-10^9 \le a_i \le 10^9$) — the elements you need to sort.

Output

In the first line print a single integer k — the number of swaps you used. Do not count the swaps outside `heapify`.

In the second line print n integers: the sorted array.

Examples

input	Copy
5 1 3 2 4 5	

output	Copy
5 1 2 3 4 5	

input	Copy
10 -1 -1 -1 -1 -1 1 1 1 1 1	

output	Copy
13 -1 -1 -1 -1 -1 1 1 1 1 1	

C. Heapsort 3 (3-ary)

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Use manual 3-ary heap implementation, do not use sets/priority queues from the standard library.

You are given n integers. Sort them using heapsort and output the number of swaps you made in `heapify`.

In this problem you are to implement 3-ary heap and 3-ary heapify function!

Use `heapify` to build a heap and then each time after you swap the head with the last element.

In `heapify`, swap only if the current element is strictly less than the greater son. Always swap with the greater son, if there are several maximal sons, swap with the first of them.

Input

The first line contains a single integer n ($1 \le n \le 10^5$).

The second line contains n integers $a_0, a_1, \ldots, a_{n-1}$ ($-10^9 \le a_i \le 10^9$) — the elements you need to sort.

Output

In the first line print a single integer k — the number of swaps you used. Do not count the swaps outside `heapify`.

In the second line print n integers: the sorted array.

Examples

input Copy

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

output Copy

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

input Copy

```
10
-1 -1 -1 -1 -1 1 1 1 1 1
```

output Copy

```
6
-1 -1 -1 -1 -1 1 1 1 1 1
```

D. Merge Arrays

time limit per test: 2.0 s
memory limit per test: 256 megabytes
input: standard input
output: standard output

`Do not use sort in this problem. You can use standard library sets/priority queues instead of heaps.`

You are given n sorted arrays of integers. Merge them in one sorted array, each element should appear the same number of times as it appears in all initial arrays together.

Input

The first line contains a single integers n ($1 \leq n \leq 10^5$) — the number of arrays.

Next n lines describe arrays. Each of them starts with an integer k ($1 \leq k \leq 10^5$) — the size of an array. Then k integers follow, each of them does not exceed 10^9 by absolute value — the elements of the array. All arrays are sorted. It is guaranteed that the total size of the arrays does not exceed 10^5.

Output

Output one line, containing the array after merging.

Example

input Copy

```
3
6 1 2 5 7 7 10
2 -1 3
4 2 6 7 11
```

output Copy

```
-1 1 2 2 3 5 6 7 7 7 10 11
```

E. k Minimums

time limit per test: 6.0 s
memory limit per test: 16 megabytes
input: standard input
output: standard output

You are given an infinite sequence of integers $a_0, a_1, a_2, \ldots$. You are also given three integers l, r and k. You have to find k minimum integers among $a_l, a_{l+1}, \ldots, a_r$, including repeats.

To get the sequence, you are given a_0, a_1 and integers A, B, C and M. For all $i \geq 2$, $a_i = (A \cdot a_{i-2} + B \cdot a_{i-1} + C) \bmod M$, where $\bmod$ is modulo operation.

Input

The first line contains six integers a_0, a_1, A, B, C and M ($0 \leq a_0, a_1, A, B, C < M$, $1 \leq M \leq 2 \cdot 10^9$).

The second line contains three integers l, r and k ($0 \leq l \leq r \leq 10^8$, $1 \leq k \leq 100$, $r - l + 1 \geq k$).

Output

Output k minimums in sorted order.

Examples

input Copy

```
2 5 3 7 3 13
2 8 3
```

output Copy

```
1 3 3
```

input Copy

```
1 1 0 1 0 7
0 7 8
```

output Copy

```
1 1 1 1 1 1 1 1
```

Note

In the first sample the sequence starts with $[2, 5, 5, 1, 12, 12, 6, 3, 3, \ldots]$.

F. Heap And Modify

time limit per test: 8.0 s
memory limit per test: 256 megabytes
input: standard input
output: standard output

`Use heap for this problem.`

In this problem you have an array, and you need to support two operations:

- Tell the current maximum of all the elements.
- Modify an element on a given position.

Input

The first line contains an integer n ($1 \leq n \leq 10^6$) — the size of the array.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($-10^9 \leq a_i \leq 10^9$) — the initial elements of the array.

The third line contains an integer m ($1 \leq m \leq 10^6$) — the number of operations.

The next m lines describe operations. Each of them starts with an integer t ($t = 1$ or $t = 2$) — the type of the operation. If $t = 1$, then you should print current maximum. Otherwise, two integers id and x follow ($1 \leq id \leq n$, $-10^9 \leq x \leq 10^9$), that means, that you should modify a_{id} to x.

Output

For each operation of type 1, print the current maximum.

Example

input Copy

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

output Copy

```
5
7
4
```

G. Segments Coloring

time limit per test: 2.0 s
memory limit per test: 256 megabytes
input: standard input
output: standard output

Use the approach discussed on lecture. Use heap to store maximum (do not use sets or other standard library data structures).

You are given a segment of length L on the Ox axis (from 0 to L). Initially, the whole segment is colored into color 0. N times some segment on it is painted: you are given its left and right ends and the new color (the ends are given by coordinates).

Your task is to tell the final color of each subsegment of length 1 (the subsegments are numbered from left to right).

The segments are colored one after another, from the first to the last. The segments can intersect and fold arbitrary.

Input

The first line contains two integers L and N ($1 \le L \le 100000$; $1 \le N \le 100000$).

N lines follow, each containing three integers l, r and c ($0 \le l \le r \le L$, $0 \le c \le 1000$): the coordinates of the left end and the right end and the color of the new segment, in that order.

Initially, the whole segment is colored into color 0.

Output

Print L integers — the final colors of the segment.

Examples

input Copy

```
5 2
0 3 1
2 4 2
```

output Copy

```
1 1 2 2 0
```