B. Restore the Permutation by Merger

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

A permutation of length n is a sequence of integers from 1 to n of length n containing each number exactly once. For example, $[1]$, $[4, 3, 5, 1, 2]$, $[3, 2, 1]$ are permutations, and $[1, 1]$, $[0, 1]$, $[2, 2, 1, 4]$ are not.

There was a permutation $p[1 \dots n]$. It was merged with itself. In other words, let's take two instances of p and insert elements of the second p into the first maintaining relative order of elements. The result is a sequence of the length $2n$.

For example, if $p = [3, 1, 2]$ some possible results are: $[3, 1, 2, 3, 1, 2]$, $[3, 3, 1, 1, 2, 2]$, $[3, 1, 3, 1, 2, 2]$. The following sequences are not possible results of a merging: $[1, 3, 2, 1, 2, 3]$, $[3, 1, 2, 3, 2, 1]$, $[3, 3, 1, 2, 2, 1]$.

For example, if $p = [2, 1]$ the possible results are: $[2, 2, 1, 1]$, $[2, 1, 2, 1]$. The following sequences are not possible results of a merging: $[1, 1, 2, 2]$, $[2, 1, 1, 2]$, $[1, 2, 2, 1]$.

Your task is to restore the permutation p by the given resulting sequence a. It is guaranteed that the answer **exists and is unique**.

You have to answer t independent test cases.

Input

The first line of the input contains one integer t $(1 \le t \le 400)$ — the number of test cases. Then t test cases follow.

The first line of the test case contains one integer n $(1 \le n \le 50)$ — the length of permutation. The second line of the test case contains $2n$ integers $a_1, a_2, \dots, a_{2n}$ $(1 \le a_i \le n)$, where a_i is the i-th element of a. It is guaranteed that the array a represents the result of merging of some permutation p with the same permutation p.

Output

For each test case, print the answer: n integers $p_1, p_2, \dots, p_n$ $(1 \le p_i \le n)$, representing the initial permutation. It is guaranteed that the answer **exists and is unique**.

Example

input Copy

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

output Copy

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