Problem 1

For a sorted singly linked list, implement the following functions

1. *Node* deleteDuplicate(Node* head)*

that removes all the duplicate values from the linked list such that every existent value should be present no more than once after this operation(Check sample cases for clarity)

2. *Node* Insert(Node* head, int val)*

which inserts a node with the given value at an appropriate position, irrespective of whether it already exists or not.

Input
First line contains a number T indicating the number of test cases
For each test case
First line contains space separated integers representing values of nodes of the linked list, **until a 0(zero) is encountered. 0 is not counted in the linked list**. The values will be sorted in ascending order.

1. Next Q lines contain queries as follows.
    a. Insert <element>
    b. Delete
    c. Print <index>

Insert will be followed by an integer which is to be inserted from the list as per the operation.
Print will be followed by an integer which is the index in the linked list which has to be printed (consider 0 indexing)

Output
For each of the test cases, print Q lines(answer to each query) as follows-:
➔ For an insertion operation, print the number just inserted into the list.
➔ For a deletion operation, print all the numbers just deleted from the list, in the order of deletion. If there are no values to remove, print “-1” (without the quotes).
➔ For a print operation, print the element at index specified (0 indexing). If an element does not exist at that index of the linked list, print “-1”.

Constraint:
$1 \leq T \leq 10$
Starting length of the list will be always >=1

$1 \leq$ Value of nodes of the linked list $\leq 10^6$

$1 \leq Q \leq 10^6$

Sample Input

```
2
12 45 47 47 58 60 60 0
7
Insert 11
Delete
Delete
Print 4
Insert 58
Insert 61
Print 13
4 5 6 7 8 9 9 9 0
5
Insert 11
Delete
Print 0
Insert 6
Print 5
```

Sample Output

```
11
47 60
-1
58
58
61
-1
11
9 9
4
6
8
```