

Day 24: More Linked Lists

Objective

Check out the [Tutorial](#) tab for learning materials and an instructional video!

Task

A Node class is provided for you in the editor. A Node object has an integer data field, *data*, and a Node instance pointer, *next*, pointing to another node (i.e.: the next node in a list).

A removeDuplicates function is declared in your editor, which takes a pointer to the *head* node of a linked list as a parameter. Complete removeDuplicates so that it deletes any duplicate nodes from the list and returns the head of the updated list.

Note: The *head* pointer may be null, indicating that the list is empty. Be sure to reset your *next* pointer when performing deletions to avoid breaking the list.

Input Format

You do not need to read any input from stdin. The following input is handled by the locked stub code and passed to the removeDuplicates function:

The first line contains an integer, *N*, the number of nodes to be inserted.

The *N* subsequent lines each contain an integer describing the *data* value of a node being inserted at the list's tail.

Constraints

- The data elements of the linked list argument will always be in non-decreasing order.

Output Format

Your removeDuplicates function should return the head of the updated linked list. The locked stub code in your editor will print the returned list to stdout.

Sample Input

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

Sample Output

```
1 2 3 4
```

Explanation

$N = 6$, and our non-decreasing list is $\{1, 2, 2, 3, 3, 4\}$. The values **2** and **3** both occur twice in the list, so we remove the two duplicate nodes. We then return our updated (ascending) list, which is $\{1, 2, 3, 4\}$.