3217. Delete Nodes From Linked List Present in Array

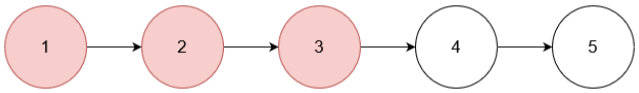
You are given an array of integers `nums` and the `head` of a linked list. Return the `head` of the modified linked list after **removing** all nodes from the linked list that have a value that exists in `nums`.

Example 1:

Input: nums = [1,2,3], head = [1,2,3,4,5]

Output: [4,5]

Explanation:



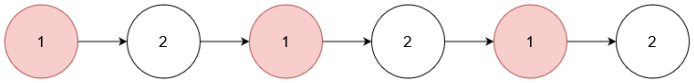
Remove the nodes with values 1, 2, and 3.

Example 2:

Input: nums = [1], head = [1,2,1,2,1,2]

Output: [2,2,2]

Explanation:



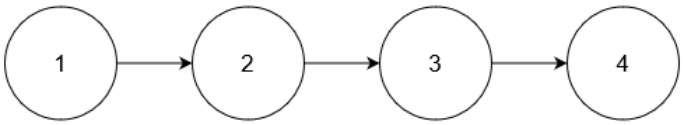
Remove the nodes with value 1.

Example 3:

Input: nums = [5], head = [1,2,3,4]

Output: [1,2,3,4]

Explanation:



No node has value 5.

Constraints:

- `1 <= nums.length <= 10^5`
- `1 <= nums[i] <= 10^5`
- All elements in `nums` are unique.
- The number of nodes in the given list is in the range `[1, 10^5]`.
- `1 <= Node.val <= 10^5`
- The input is generated such that there is at least one node in the linked list that has a value not present in `nums`.

3217. Delete Nodes From Linked List Present in Array

```
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode() : val(0), next(nullptr) {}
 *     ListNode(int x) : val(x), next(nullptr) {}
 *     ListNode(int x, ListNode *next) : val(x), next(next) {}
 * };
 */
```

3217. Delete Nodes From Linked List Present in Array

```
/*
    Sorting+Binary search (Memory leaks)
    if head... else...

    Time complexity: O(nlogn)
    Space complexity: O(logn)
*/
typedef std::vector<int> vi;

class Solution {
    public:
        ListNode* modifiedList(vi& nums, ListNode* head){
            std::sort(nums.begin(),nums.end());

            ListNode* fast=head;
            ListNode* slow=head;
            while(fast){
                if(std::binary_search(nums.begin(),nums.end(),fast->val)){
                    if(fast==head) {
                        head=head->next;
                        fast=slow=head;
                     }
                    else{
                        slow->next=fast->next;
                        fast=fast->next;
                     }
                }
                else{
                    slow=fast;
                    fast=fast->next;
                }
            }
            return head;
        }
};
```

3217. Delete Nodes From Linked List Present in Array

```
/*
    Sorting+Binary search (Memory leaks)
    Use dummy node to make a unified pattern to all nodes

    Time complexity: O(nlogn)
    Space complexity: O(logn)
*/
typedef std::vector<int> vi;
class Solution {
   public:
      ListNode* modifiedList(vi& nums, ListNode* head){
         std::sort(nums.begin(),nums.end());

         ListNode* dummy=new ListNode(0,head);
         ListNode* fast=head;
         ListNode* slow=dummy;
         while(fast){
            if(std::binary_search(nums.begin(),nums.end(),fast->val)) slow->next=fast->next;
            else  slow=fast;
            fast=fast->next;
         }
         return dummy->next;
      }
};
```

3217. Delete Nodes From Linked List Present in Array

```
/*
    Sorting+Binary search (No memory leaks)
    Use dummy node to make a unified pattern to all nodes

    Time complexity: O(nlogn)
    Space complexity: O(logn)
*/
typedef std::vector<int> vi;
class Solution {
    public:
        ListNode* modifiedList(vi& nums, ListNode* head){
            std::sort(nums.begin(),nums.end());

            ListNode* dummy=new ListNode(0,head);
            ListNode* fast=head;
            ListNode* slow=dummy;
            ListNode* to_delete=nullptr;

            while(fast){
                if(std::binary_search(nums.begin(),nums.end(),fast->val)){
                    slow->next=fast->next;
                    to_delete=fast;
                }
                else slow=fast;

                fast=fast->next;
                delete to_delete;
                to_delete=nullptr;
            }

            head=fast=slow=to_delete=nullptr;

            return dummy->next;
        }
};
```

3217. Delete Nodes From Linked List Present in Array

```
/*
    bitset (No memeory leaks)
    Use dummy node to make a unified pattern to all nodes

    Time complexity: O(n)
    Space complexity: O(10^5)
*/
typedef std::vector<int> vi;
class Solution {
    public:
        ListNode* modifiedList(vi& nums, ListNode* head){
            std::bitset<100001> to_delete=0;
             for(auto& e: nums) to_delete[e]=1;

            ListNode* dummy=new ListNode(0,head);
            ListNode* fast=head;
            ListNode* slow=dummy;
            ListNode* to_delete_ptr=nullptr;

            while(fast){
                if(to_delete[fast->val]){
                    slow->next=fast->next;
                    to_delete_ptr=fast;
                }
                else slow=fast;

                fast=fast->next;
                delete to_delete_ptr;
                to_delete_ptr=nullptr;
            }

            head=fast=slow=to_delete_ptr=nullptr;

            return dummy->next;
        }
};
```