



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

Experiment-1.2

Student Name: Nabha Varshney

UID: 20BCS4995

Branch: CSE

Section/Group: 20BCS-DM-704 (A)

Semester: 6th

Date of Performance: 22th Feb 2023

Subject Name: Competitive Coding II

Subject Code: 20CSP- 351

Aim – To implement the concept of Arrays, Queues and Stack and Linked List.

Objective-

- ♦ The objective is to build problem solving capability and to learn the basic concepts of data structures.
- ♦ The implementation of arrays, queues which shows and brushes up the concept of 1D, 2D arrays and can be solved through various approaches.
- ♦ The implementation of removing duplicates from the sorted list was introduced.

1) Jump Game II

<https://leetcode.com/problems/jump-game-ii/>

Code –

```
class Solution {
public:
    int jump(vector<int>& nums) {
        int len=nums.size()-1;
        int curr=-1,next=0,ans=0;
        for(int i=0;next<len;i++)
        {
            if(i>curr){
                ans++;
                curr=next;
            }
        }
    }
};
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
    }  
    next=max(next,nums[i]+1);  
    }  
    return ans;  
}  
};
```

Output -

LeetCode

45. Jump Game II

Medium

You are given a 0-indexed array of integers `nums` of length `n`. You are initially positioned at `nums[0]`.

Each element `nums[i]` represents the maximum length of a forward jump from index `i`. In other words, if you are at `nums[i]`, you can jump to any `nums[i + j]` where:

- $0 \leq j \leq \text{nums}[i]$ and
- $i + j < n$

Return the minimum number of jumps to reach `nums[n - 1]`. The test cases are generated such that you can reach `nums[n - 1]`.

Example 1:

Input: `nums = [2,3,1,1,4]`
Output: 2
Explanation: The minimum number of jumps to reach the last index is 2. Jump 1 step from index 0 to 1, then 3 steps to the last index.

Example 2:

Input: `nums = [2,3,0,1,4]`
Output: 2

```
class Solution {  
public:  
    int jump(vector<int>& nums) {  
        int len=nums.size()-1;  
        int curr=1, next=0, ans=0;  
        for(int i=0; i<len; i++)  
        {  
            if(i>curr){  
                ans++;  
                curr=next;  
            }  
            next=max(next, nums[i]+1);  
        }  
        return ans;  
    }  
};
```

Testcase Result

Input

`nums =`
`[2,3,1,1,4]`

Output

2

Expected

2

Console Run Submit

2) Remove the duplicate elements from list

<https://leetcode.com/problems/remove-duplicates-from-sorted-list-ii/>

Code -

```
class Solution {
public:
    ListNode* deleteDuplicates(ListNode* head) {

        if(head==0 || head->next==0)return head;
        int flag=0;
        ListNode*prev=0,*cur=head,*nex=head->next,*dummy=0;
        prev=new ListNode;
        dummy=prev;
        prev->next=head;
        while(nex!=0)
        {
            if(cur->val==nex->val)
            {
                prev->next=nex;
                cur=prev->next;
                nex=cur->next;
                flag=1;
            }
            else if(flag==1)
            {
                prev->next=nex;
                cur=prev->next;
                nex=cur->next;
                flag=0;
            }
            else
            {
                prev=prev->next;
                cur=cur->next;
                nex=nex->next;
            }
        }
    }
}
```

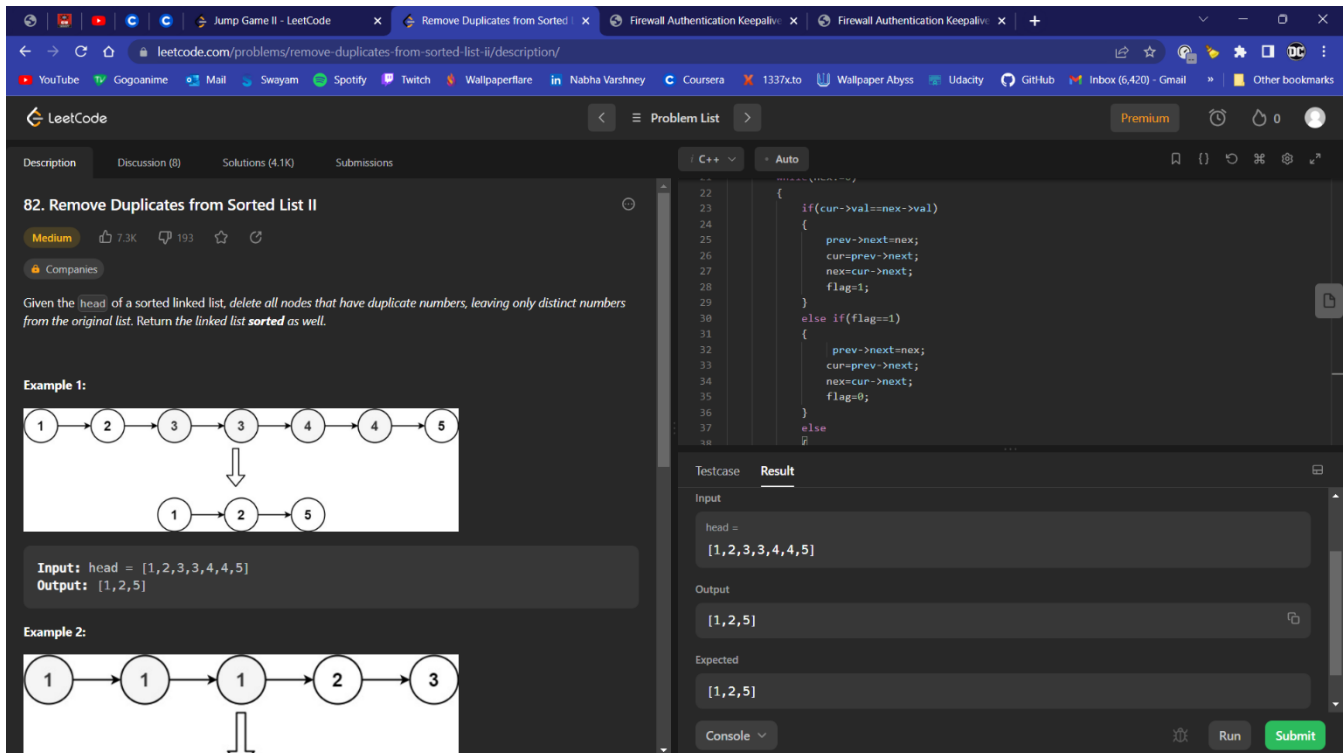
```

        if(flag==1)
        {
            prev->next=0;
        }
        return dummy->next;

    }
};

```

Output -



82. Remove Duplicates from Sorted List II

Medium 7.3K 193

Companies

Given the *head* of a sorted linked list, delete all nodes that have duplicate numbers, leaving only distinct numbers from the original list. Return the linked list *sorted* as well.

Example 1:

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

Example 2:

Input: head = [1,1,1,2,3]
Output: [2,3]

```

22  {
23      if(cur->val==nex->val)
24      {
25          prev->next=nex;
26          cur=prev->next;
27          nex=cur->next;
28          flag=1;
29      }
30      else if(flag==1)
31      {
32          prev->next=nex;
33          cur=prev->next;
34          nex=cur->next;
35          flag=0;
36      }
37      else
38      {

```

Testcase Result

Input

head = [1,2,3,3,4,4,5]

Output

[1,2,5]

Expected

[1,2,5]

Console Run Submit