DEPARTMENT OF
COMPUTER SCIENCE & ENGINEERING
Discover. Learn. Empower.

CHANDIGARH UNIVERSITY
Discover. Learn. Empower.

UNIVERSITY INSTITUTE OF ENGINEERING

Department of Computer Science & Engineering

Subject Name: Competitive Coding LAB

Subject Code: 20CSP-351

Submitted to:

Er. Santosh

Submitted by:

Name: Aayush Gurung

UID: 20BCS5323

Section: DM_607

Group: A

INDEX

Ex. No	List of Experiments	Conduct (MM: 12)	Viva (MM: 10)	Record (MM: 8)	Total (MM: 30)	Remarks/Signature
1.1						
1.2						
1.3						
2.1						
2.2						
2.3						
2.4						
3.1						
3.2						
3.3						

Experiment1.1

Student Name: Aayush Gurung **UID:20BCS5323**
Branch: CSE **Section/Group:DM_607(A)**
Semester: 6 **Date of Performance:17-02-2023**
Subject Name: CC LAB **Subject Code: 20CST-355**

Aim:

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

Objective

Problem 1:
Implement Jump Game

Approach:
Greedy Approach.

Algorithm:

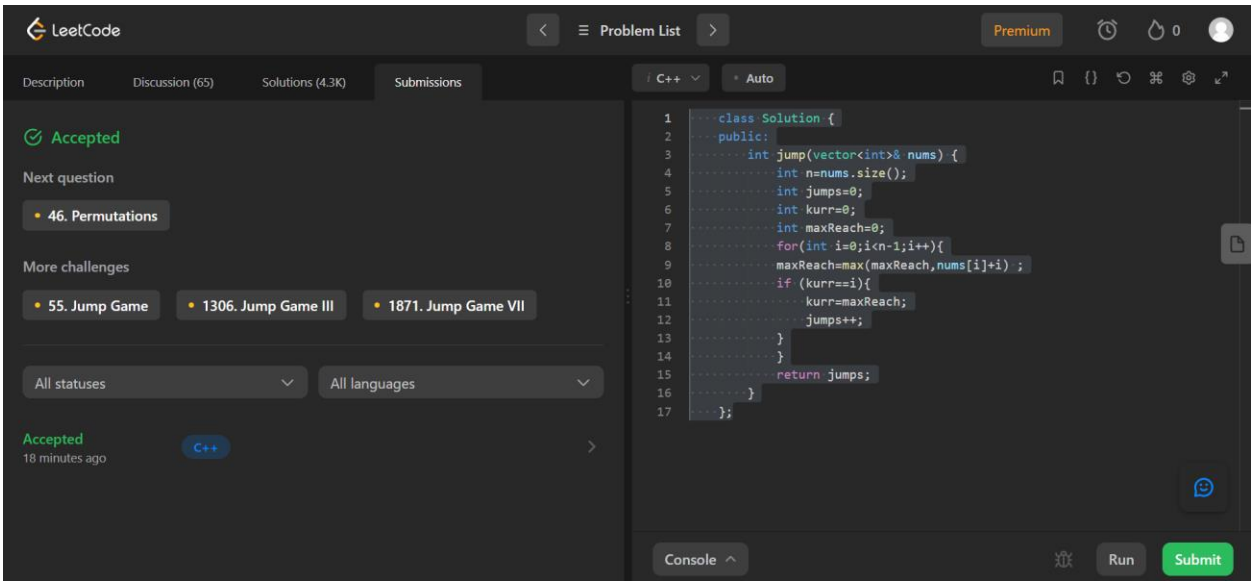
1. Declare jumps, kurr and maxReach and set them to 0.
2. Then make a loop and update maxReach as the max of maxReach and the maximum jump a particular index can take.
3. If i== kurr then update kurr =maxReach and jump++;
4. Return jumps.

Code:

```
class Solution {
public:
    int jump(vector<int>& nums) {
        int n=nums.size();
        int jumps=0;
        int kurr=0;
        int maxReach=0;
        for(int i=0;i<n-1;i++){
        maxReach=max(maxReach,nums[i]+i) ;
        if (kurr==i){
            kurr=maxReach;
            jumps++;
        }
```

```
            }
            return jumps;
        }
    };
```

Output:



Problem 2:

Remove Duplicates from Sorted List II

Approach:
If the head value and the head next value is equal then go the the last duplicate listnode and update head value to that's next, do that for all the elements in the list and then return the head at last.

Algorithm:

1. Declare 2 ListNode pointers and declare one with the head and the other as that one. One is a and the other is pre.
2. Then iterate over the list and if there's a duplicate value then put that in a while loop and head=head->next;
3. If no duplicate value then pre->next=pre->next and head=head-next;
4. Return a->next;

Code:

```
/**
 * 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) {}
 * };
 */
class Solution {
public:
    ListNode* deleteDuplicates(ListNode* head) {
       ListNode* a=new ListNode(0, head);
       ListNode* pre=a;
       while(head!=NULL){
           if(head->next!=NULL&&(head->val)==((head->next)->val)){
               while(head->next!=NULL&&(head->val)==((head->next)->val))
               head=head->next;
               pre->next=head->next;
           }
           else {
               pre=pre->next;
           }
                   head=head->next;
```

```
        }
            return a->next;

         }
};
```

Output: