EXPERIMENT - 1

Student Name: Sandeep Kumar **UID: 20BCS4885**

Branch: CSE **Section/Group: 603/A**

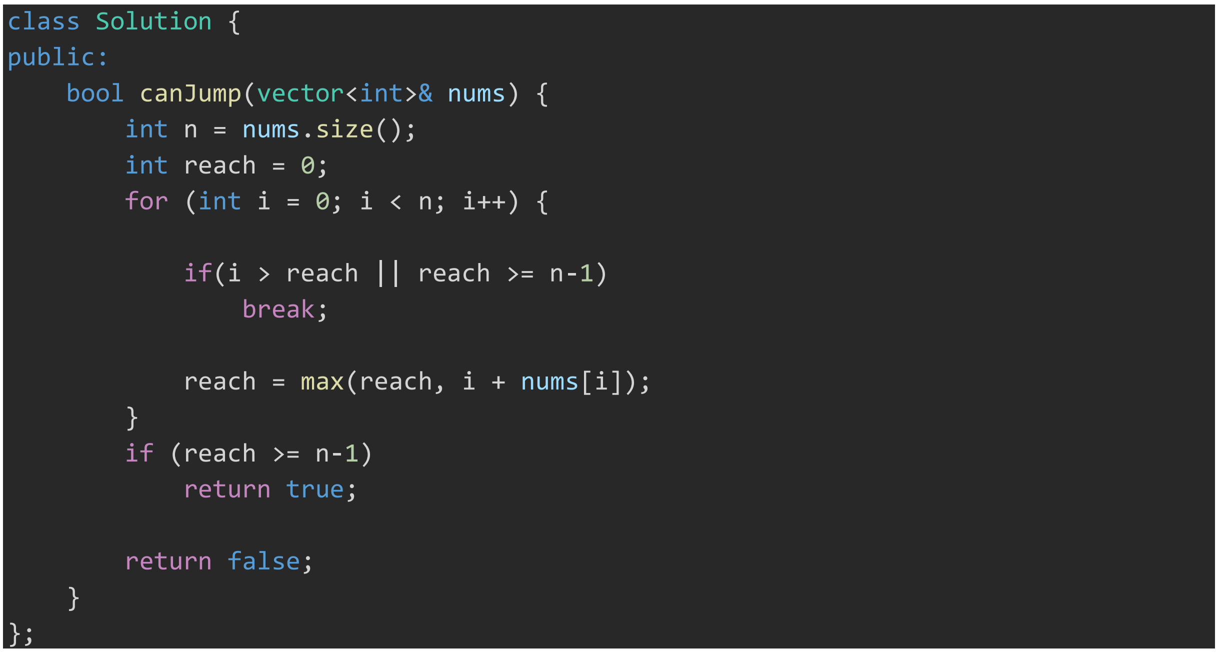
Semester: 6th semester **Subject: Competitive Coding**

Task-1: Jump Game

https://leetcode.com/problems/jump-game/

Solution:

Input code:



```
class Solution {
public:
    bool canJump(vector<int>& nums) {
        int n = nums.size();
        int reach = 0;
        for (int i = 0; i < n; i++) {

            if(i > reach || reach >= n-1)
                break;

            reach = max(reach, i + nums[i]);
        }
        if (reach >= n-1)
            return true;

        return false;
    }
};
```

Approach:

Initialize a variable reach to 0, which represents the farthest index that can be reached so far. Loop through the array nums and for each index i, do the following:

a) If i is greater than reach or reach is greater than or equal to nums.length - 1, break the loop as it means reaching the last index is not possible.
b) Update the value of reach as the maximum of reach and i + nums[i].

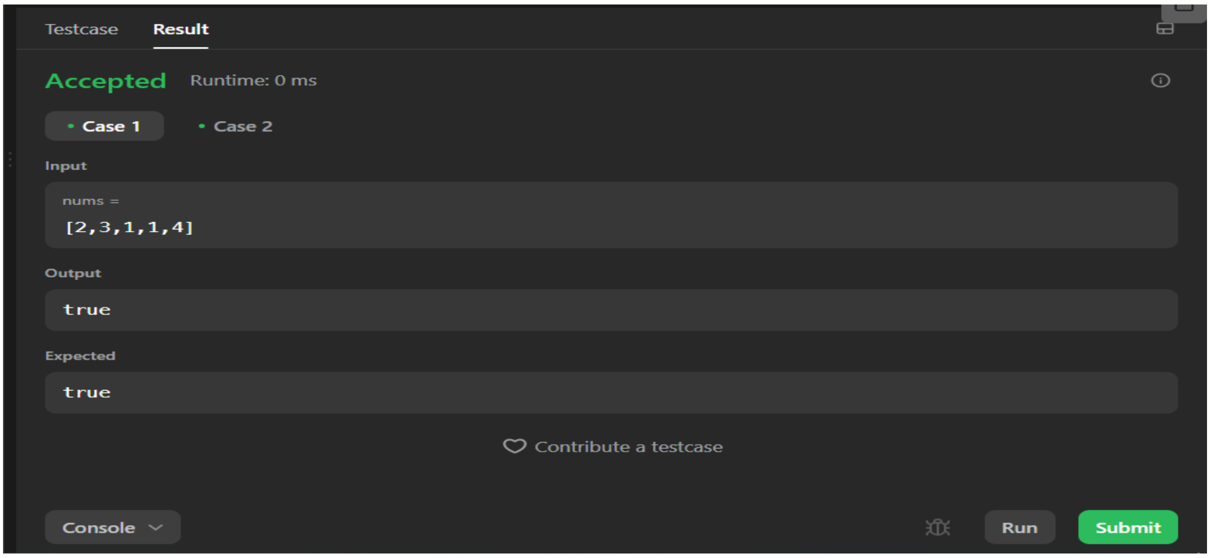
Return reach >= nums.length - 1, which means that the last index can be reached or not.

Complexity:

Time complexity:
O(n), where n is the length of the array nums. This is because we are looping through the entire nums array once

Space complexity:
O(1), as we are using a single integer variable reach.

Output:



Accepted Runtime: 0 ms

Input

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

Output

true

Expected

true

Task-2: Remove Duplicates from Sorted List.

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

Solution:

Input Code:

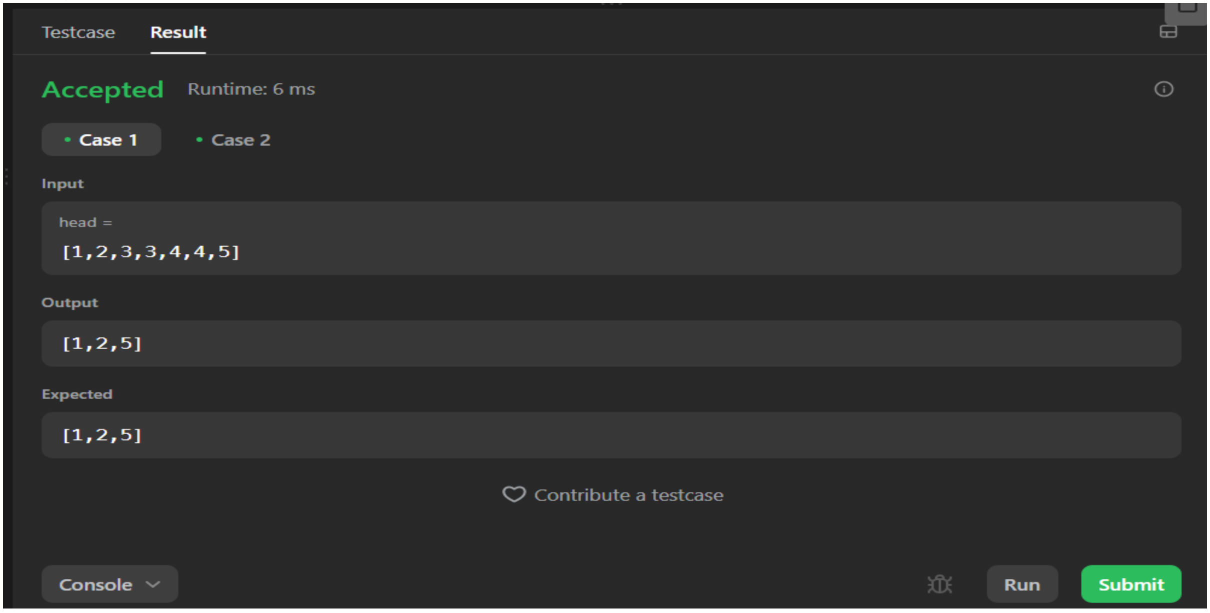
```
class Solution {
public:
    ListNode* deleteDuplicates(ListNode* head) {
        ListNode *answer = new ListNode(0);
        ListNode *curr = answer;
        while(head) {
            bool flag = false;
            while(head->next && head->val == head->next->val){
                flag = true;
                head = head->next;
            }
            if(!flag) {
                curr -> next = head;
                curr = head;
                head = head -> next;
                curr -> next = nullptr;
            } else {
                 head = head -> next;
            }
        }

        return answer->next;
    }
};
```

Complexity:

Time complexity:
O(n)

Space complexity:
O(1)

Output:



Accepted Runtime: 6 ms

Input

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

Output

```
[1,2,5]
```

Expected

```
[1,2,5]
```