**STUDENT NAME: - KINJAL CHOWDHURY**

**UID: - 20BCS3876**

**SECTION: - DWWC - 43**

**SUBJECT: - IT SKILLS**

**SUBMITTED TO :- NATASHA MA'AM**

## Q1) https://leetcode.com/problems/insertion-sort-list/

## Program code :-

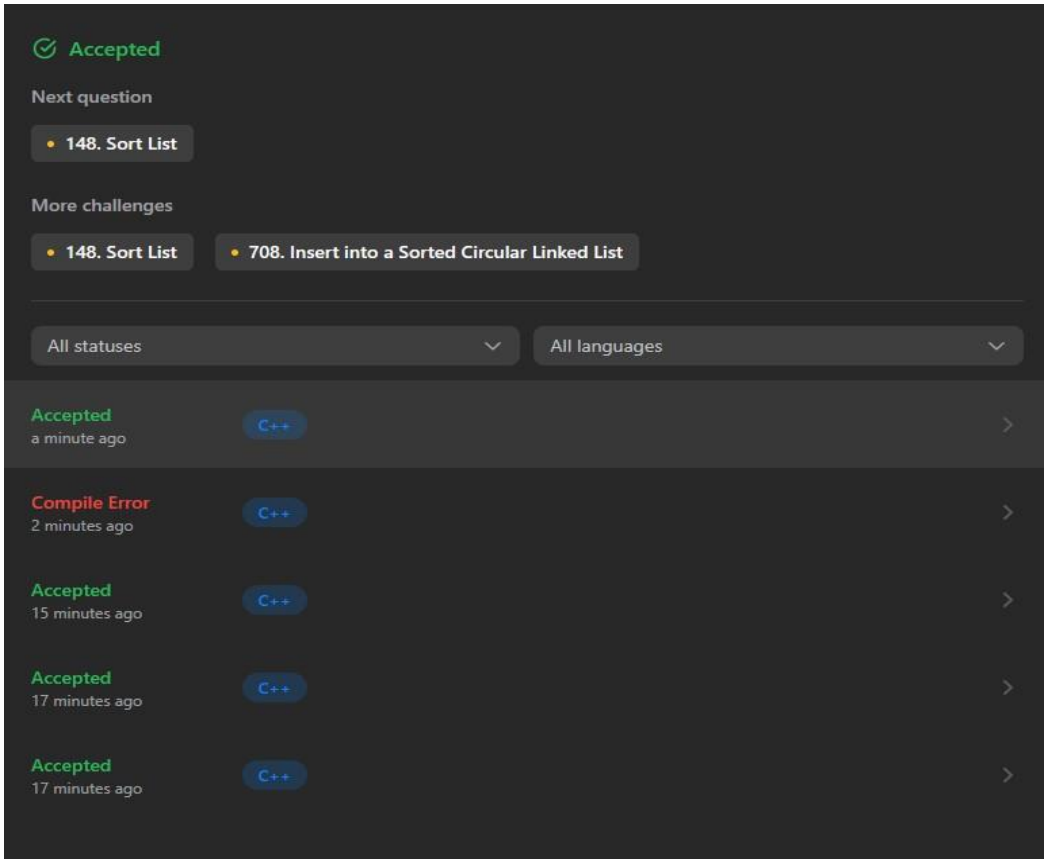
```
class Solution { public:
   ListNode* insertionSortList(ListNode* head) {
      ListNode* newHead = NULL;//initializing the newHead for our sorted linkedlist
while(head){
         // Exluding node from the original linked list we will do this one at a time
         ListNode* temp = head;          head = head-
>next;          temp->next=NULL;

         //setting the first node of our final linked list          if(newHead == NULL) newHead
= temp;
          // if the position of element is at index 0 i.e. at the start (the temp node is the smallest
of all the nodes that are currently present in the sorted linked list)          else if(newHead-
>val >= temp->val){          temp->next = newHead;          newHead = temp;
         }
         // inserting the node anywhere in the middle or in the end depending upon the
value of the temp node;          else{
            ListNode* root = newHead;          {
```

```
            while(root->next){
                if(temp->val > root->val and temp->val <= root-
>next->val){
                    temp->next = root->next;                    root->next
= temp;                    break;
                }
                root = root->next;
            }
                //inserting the temp node at the end                if(root->next==NULL) root-
>next = temp;

            }
          }
       }
       //Our sorted linkedlist        return newHead;
   }
};
```

## Output :-

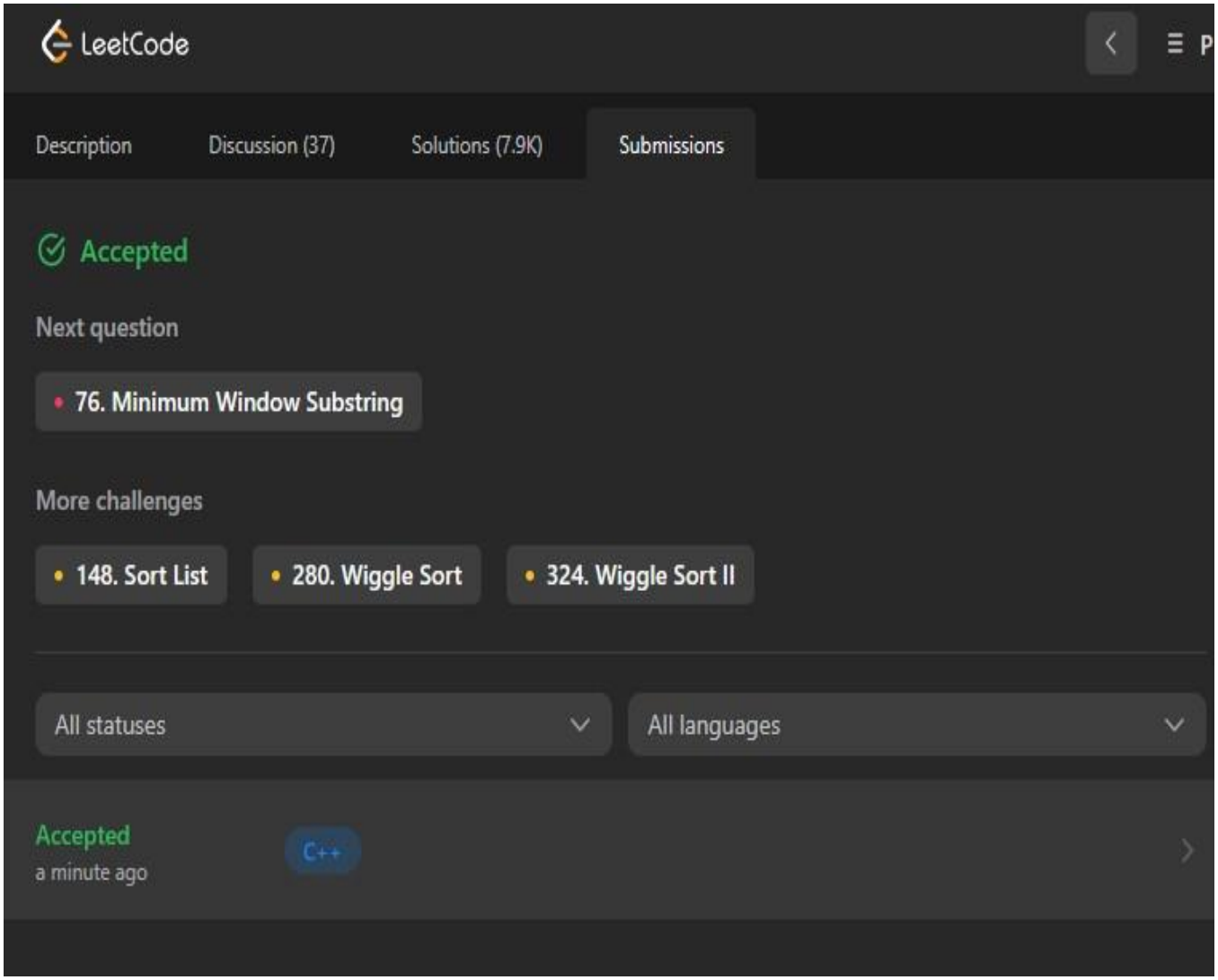


## Q2) https://leetcode.com/problems/sort-colors/

## Program code :-

```
class Solution {
public:
    void sortColors(vector<int>& nums) {
        sort(nums.begin(),nums.end());
    }
};
```

## Output :-

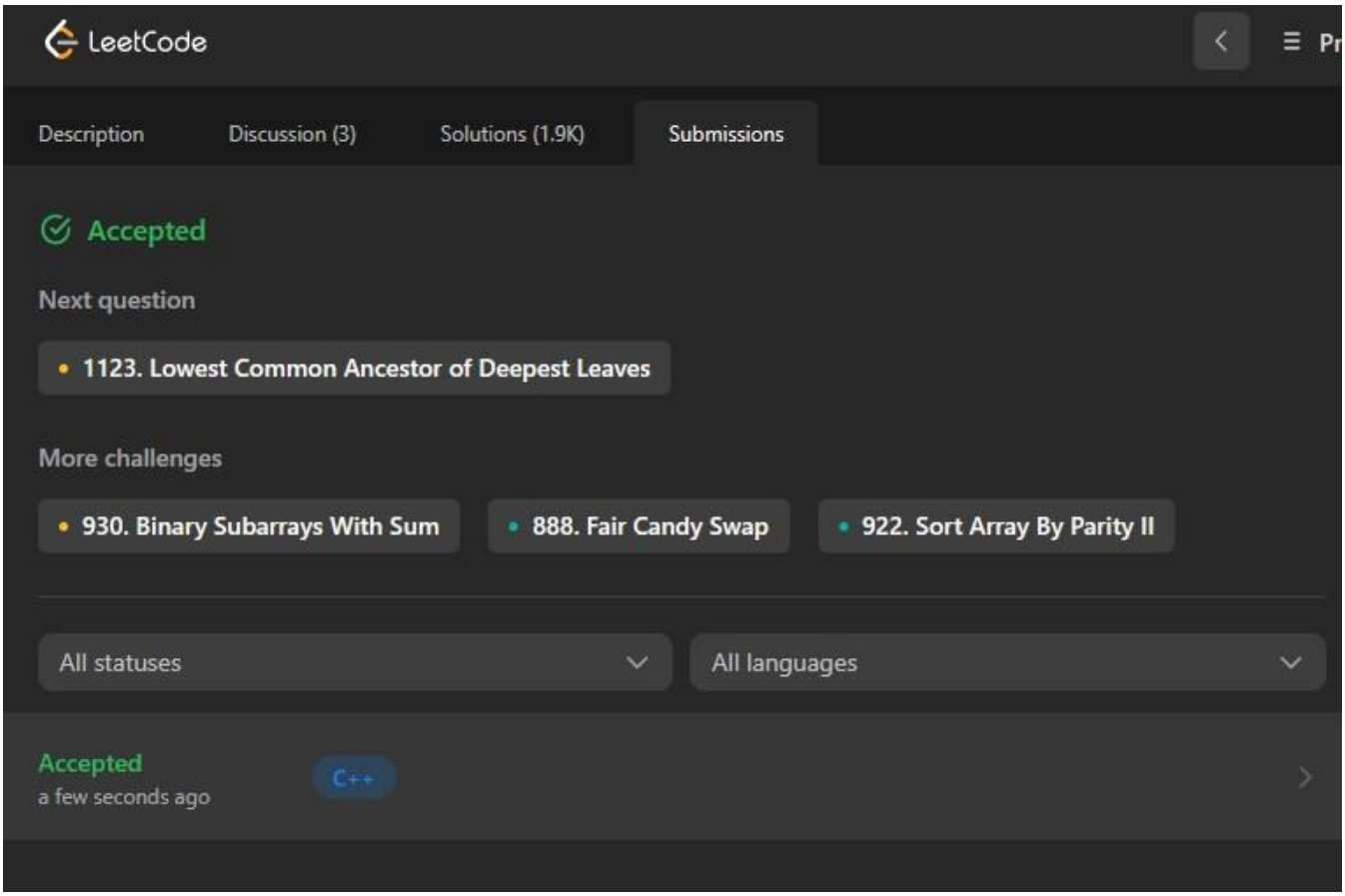


**Q3)** https://leetcode.com/problems/relative-sort-array/

## Program code :-

```
class Solution {
public:
    vector<int> relativeSortArray(vector<int>& arr1, vector<int>&
arr2) {
        map<int, int> m;
        for (auto i : arr1) m[i]++;
        int pos = 0;
        for (auto j : arr2) {
            while(m[j]-- > 0) arr1[pos++] = j;
        }
        for (auto it : m) {
            while(it.second-- > 0) arr1[pos++] = it.first;
        }
        return arr1;
    }
};
```

## Output :-

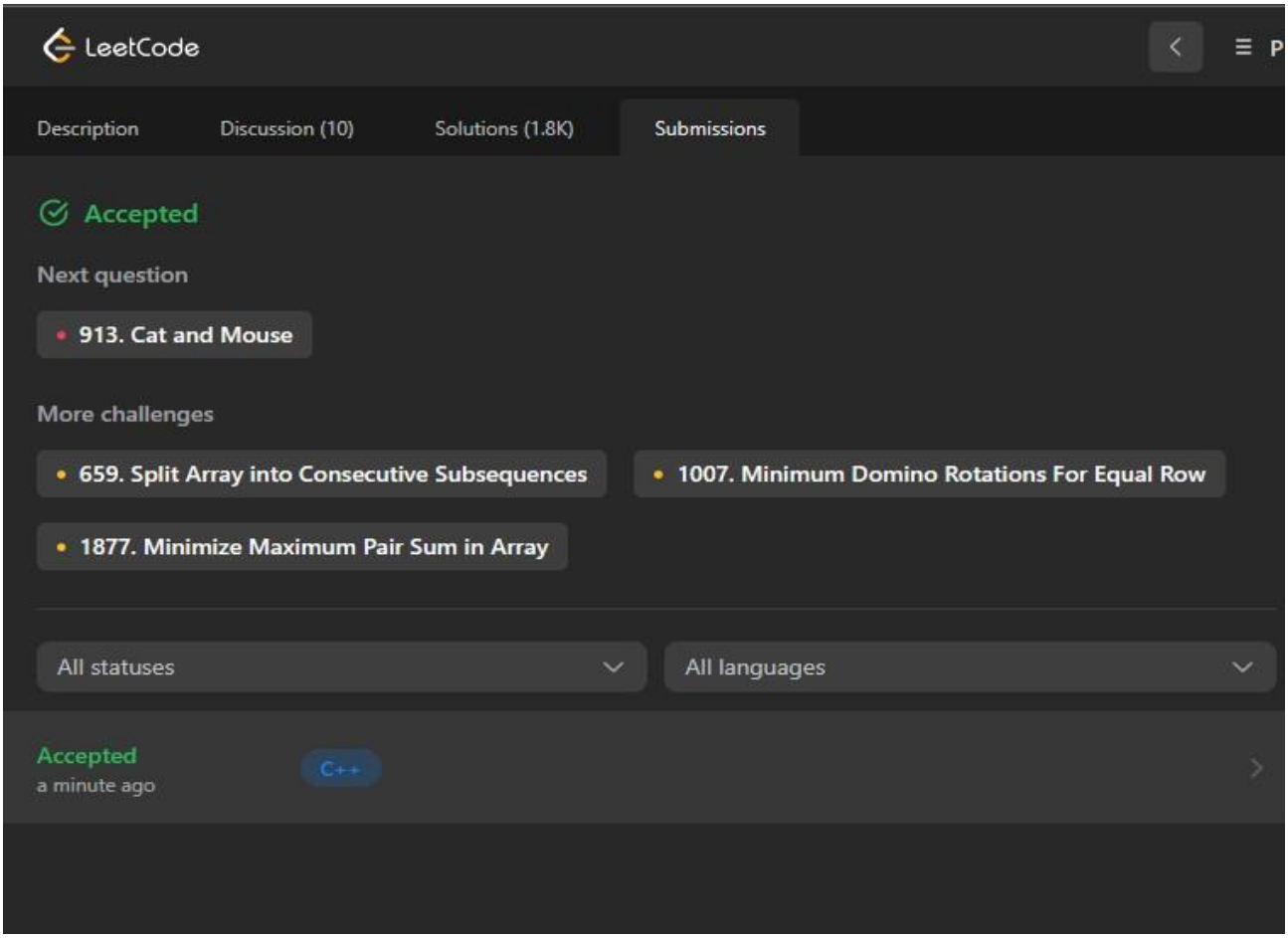



## Q4) https://leetcode.com/problems/sort-an-array/

```
class Solution {
public:
    vector<int> sortArray(vector<int>& nums) {
        priority_queue<int, vector<int>, greater<int>>q;
        for(int i=0; i<nums.size(); i++){
            q.push(nums[i]);
        }
        vector<int>ans;
        while(!q.empty()){
            ans.push_back(q.top());
```

```
            q.pop();
        }
    return ans;
    }
};
```

## Output :-



## Q5) https://www.codechef.com/submit-v2/TSORT?tab=solution

```
#include <bits/stdc++.h>
using namespace std; int
main() {
        // your code goes here
int t;    cin>>t;
        vector <int> a(t);
for(int i = 0; i< t ; i++){
cin>>a[i];
        }
        sort(a.begin(),a.end());
for(int x : a)
cout<<x<<endl;          return
0;
}
```

## Output :-