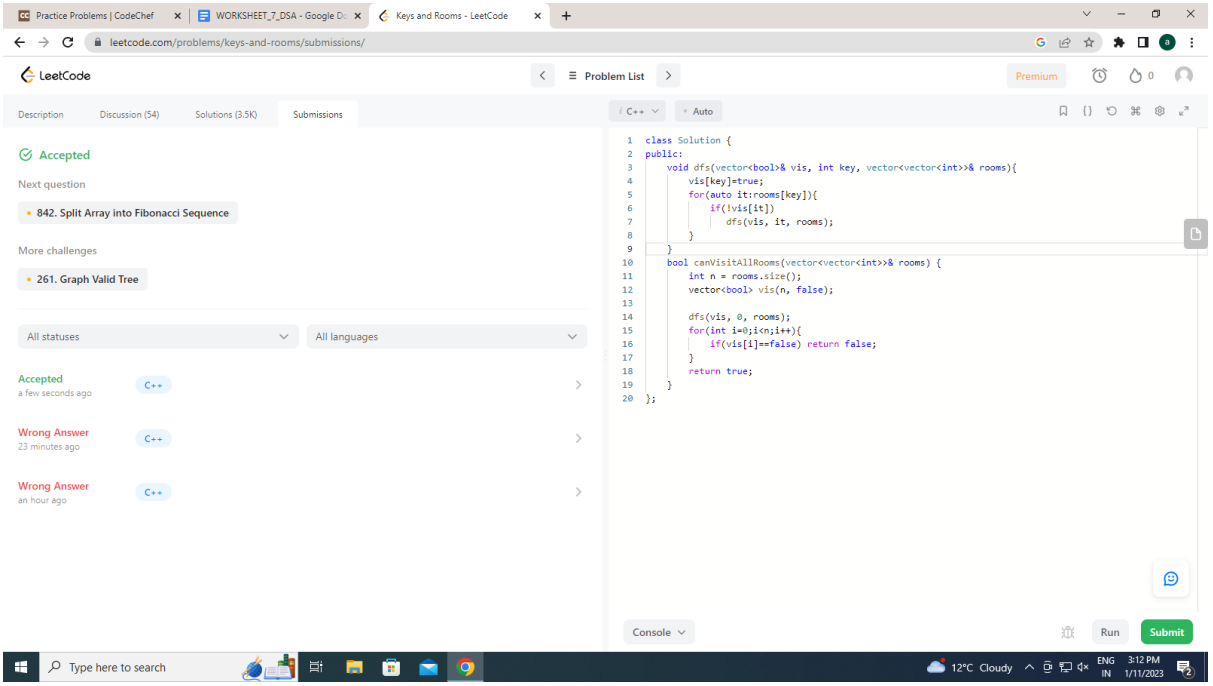
WORKSHEET_7

1. Keys and Rooms

```
class Solution {
public:
    void dfs(vector<bool>& vis, int key, vector<vector<int>>& rooms){
        vis[key]=true;
        for(auto it:rooms[key]){
            if(!vis[it])
               dfs(vis, it, rooms);
        }
    }
    bool canVisitAllRooms(vector<vector<int>>& rooms) {
        int n = rooms.size();
        vector<bool> vis(n, false);

        dfs(vis, 0, rooms);
        for(int i=0;i<n;i++){
            if(vis[i]==false) return false;
        }
        return true;
    }
};
```



2. Mango Market

```
#include <bits/stdc++.h>

using namespace std;

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);
    int n, m;
        cin >> n >> m;
        long long sum = 0;
        for (int i = 1; i <= n; i++) {
            long long x;
            cin >> x;
            sum += x;
        }
        long long edges = (long long)m, unused = ((long long)n * (n - 1)) / 2LL - edges;
        for (int i = 0; i < m; i++) {
            int u, v;
            cin >> u >> v;
        }
        int b=edges-unused;
        int q;
        cin >> q;
        for (int i = 0; i < q; i++) {
            char x;
            cin >> x;
            if (x == '?') {
                cout << sum + edges-unused << '\n';
                continue;
            }
            int u, v;
                cin >> u >> v;
            if (x == '+') {
              edges++;unused--;

            }
            else if (x == '-') {
              edges--;
              unused++;
            }

        }
        return 0;
```

}

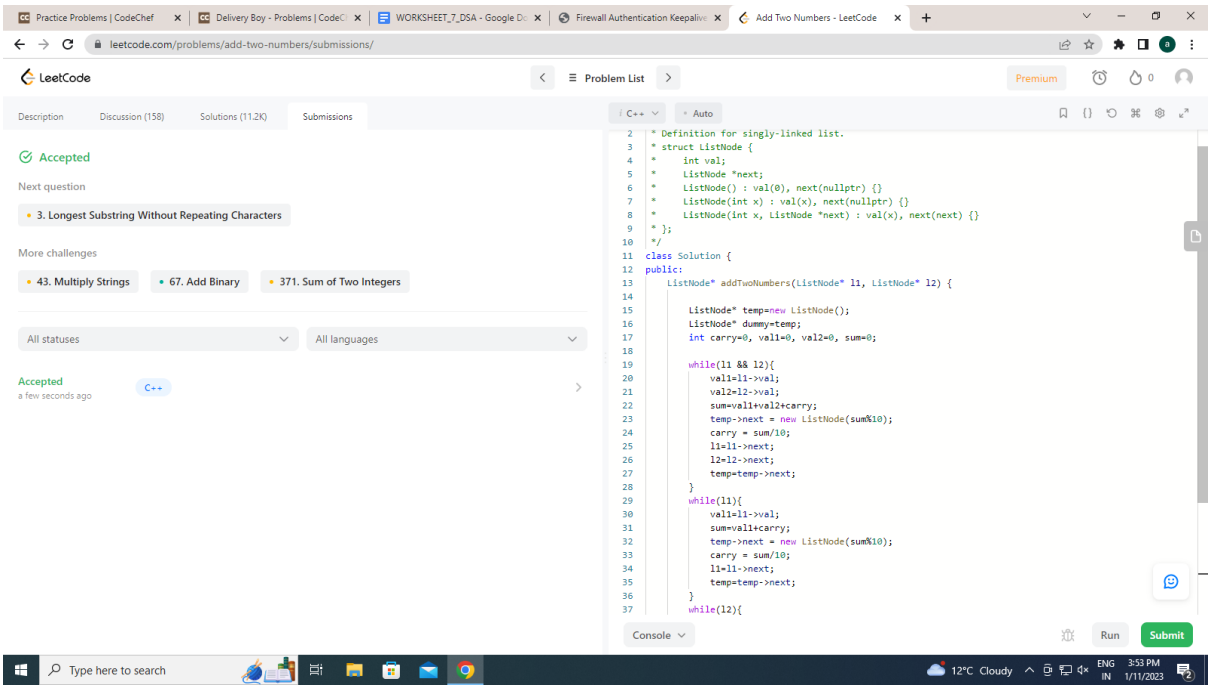
3. Add Two Numbers

```
/**
 * 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* addTwoNumbers(ListNode* l1, ListNode* l2) {

        ListNode* temp=new ListNode();
        ListNode* dummy=temp;
        int carry=0, val1=0, val2=0, sum=0;

        while(l1 && l2){
            val1=l1->val;
            val2=l2->val;
            sum=val1+val2+carry;
            temp->next = new ListNode(sum%10);
            carry = sum/10;
            l1=l1->next;
            l2=l2->next;
            temp=temp->next;
        }
        while(l1){
            val1=l1->val;
            sum=val1+carry;
            temp->next = new ListNode(sum%10);
            carry = sum/10;
            l1=l1->next;
            temp=temp->next;
        }
        while(l2){
            val2=l2->val;
            sum=val2+carry;
```

```
            temp->next = new ListNode(sum%10);
            carry = sum/10;
            l2=l2->next;
            temp=temp->next;
        }
        if(carry) temp->next=new ListNode(carry);
        return dummy->next;
    }
};
```



4. Chef and Reversing

```
#include <bits/stdc++.h>
using namespace std;
const int N = 1e5+10;
const int infi=1e9+10;
vector<pair<int,int>>g[N];
vector<int>level(N,infi);
int n,m;
 void bfs(){
    level[1]=0;
    deque<int> dq;
    dq.push_back(1);
    while(!dq.empty()){
        int cur_v= dq.front();
```

```
        dq.pop_front();
        for(auto childs:g[cur_v]){
            int child = childs.first;
            int wt = childs.second;
            if(level[cur_v]+wt < level[child]){
            level[child] = level[cur_v] + wt;
            if(wt==1) dq.push_back(child);
            else dq.push_front(child);
        }}
    }
  if(level[n]==infi) cout<<-1 ;
  else cout<<level[n];
}


int main() {

    cin>>n>>m;
    for(int i=0;i<m;i++){
        int x,y;
        cin>>x>>y;
        if(x==y)continue;
        g[x].push_back({y,0});
        g[y].push_back({x,1});
    }
    bfs();

    return 0;
}
```

Minimal Travel Time

```
#include <bits/stdc++.h>

#define llint long long int
using namespace std;
```

```
void run()
{
    // Insert code here
    int n, m, s, k;
    cin >> n >> m >> s >> k;

    vector<vector<int>> graph(n+1);

    for(int i = 0; i < m; ++i){
        int u, v;
        cin >> u >> v;
        graph[u].push_back(v);
        graph[v].push_back(u);
    }

    std::vector<int> count(n+1);
    for (int i = 0; i < s; ++i){
        int val;
        cin >> val;
        count[val]++;
    }
    vector<bool> vis(n+1);
    queue<int> q;

    q.push(0);
    vis[0] = true;

    llint res = 0, curr = 0;

    while(!q.empty() && k > 0){
        int size = q.size();
        for(int i = 0; i < size; ++i){
            int node = q.front();
            q.pop();
            for(auto adj : graph[node]){
                if(!vis[adj]){
                    vis[adj] = true;
                    q.push(adj);
                }
            }
            int val = min(k, count[node]);
            res += 2*curr*val;
            k -= val;
        }
        curr++;
    }
    cout << res << "\n";
}
```

```
int main()
{
    std::ios_base::sync_with_stdio(false);
    std::cin.tie(NULL);

    int t = 1;
    std::cin >> t;
    while (t--)
        run();

    return 0;
}
```