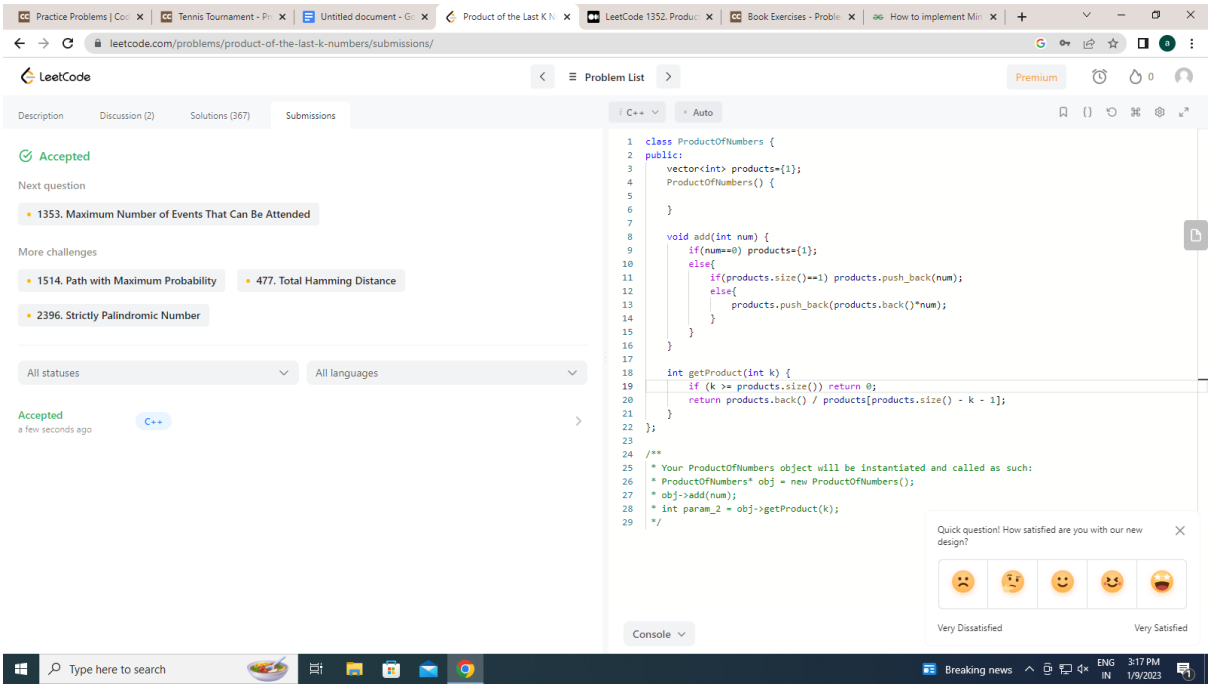
WORKSHEET_5

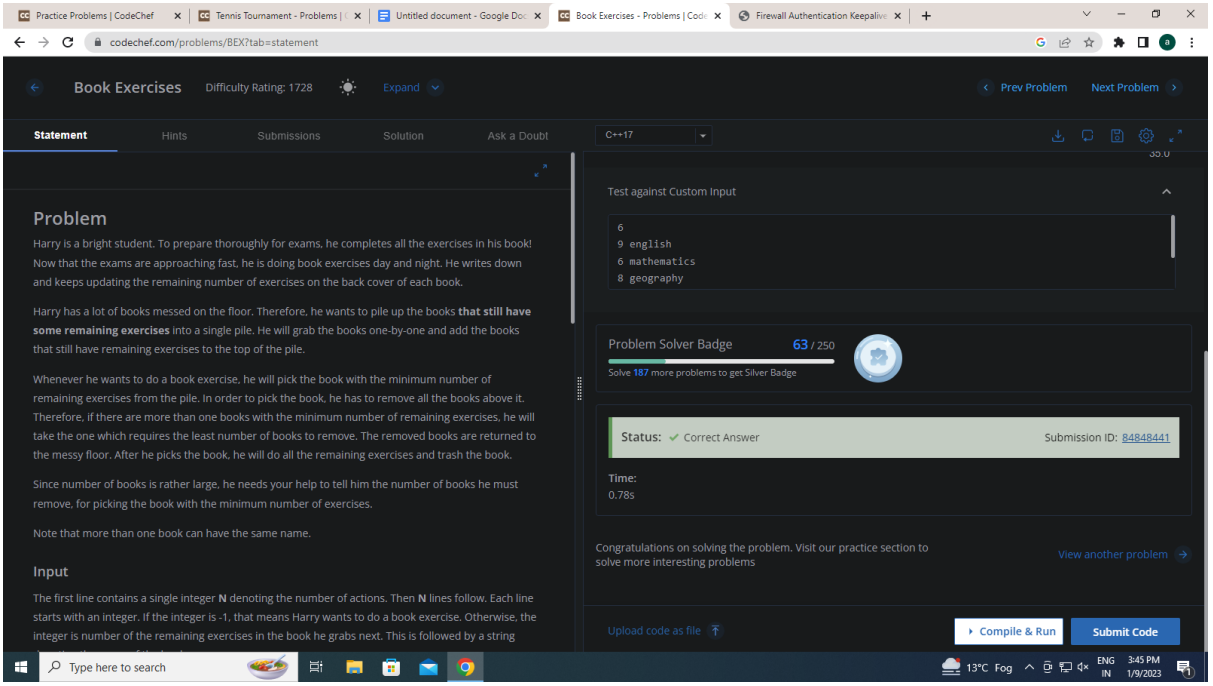
1. Product of the Last K Numbers

```
vector<int> products={1};
    ProductOfNumbers() {

    }

    void add(int num) {
        if(num==0) products={1};
        else{
            if(products.size()==1) products.push_back(num);
            else{
                products.push_back(products.back()*num);
            }
        }
    }

    int getProduct(int k) {
        if (k >= products.size()) return 0;
        return products.back() / products[products.size() - k - 1];
    }
```



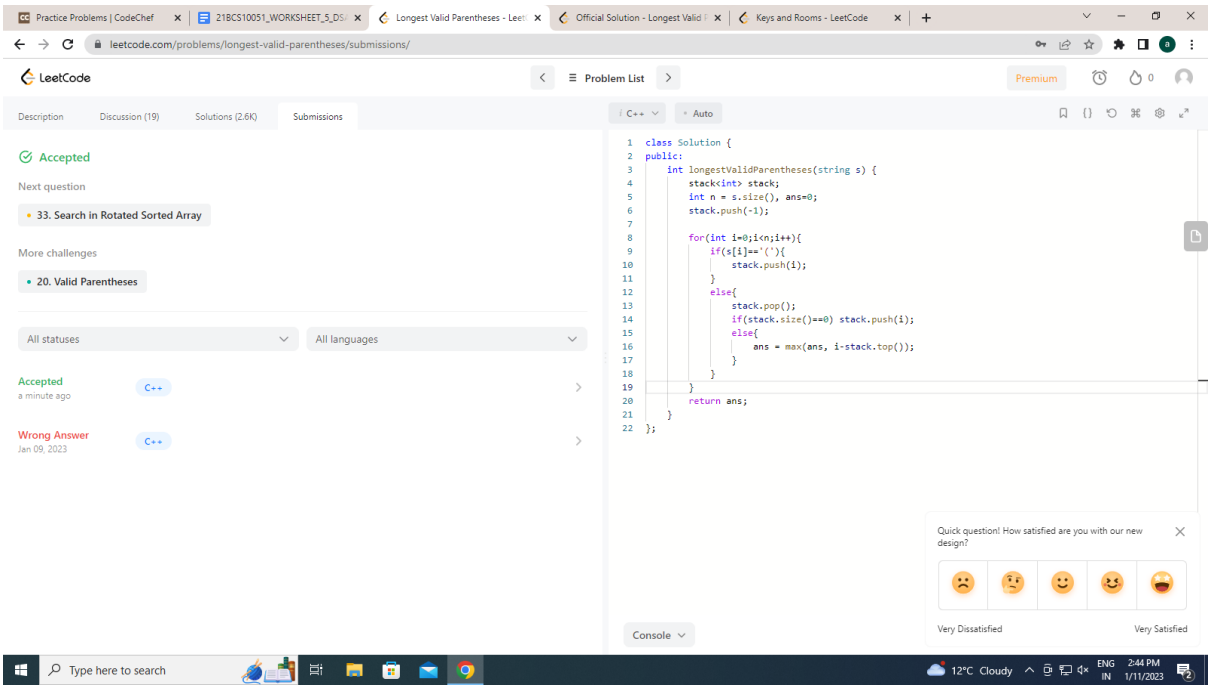
2. Book Exercises

```
#include <iostream>
#include<bits/stdc++.h>
using namespace std;

int main() {
ios_base::sync_with_stdio(false);
    cin.tie(NULL);
            int n;
            cin>>n;
            stack<pair<pair<int,string>,int>> st1;
            for(int i=0;i<n;i++){
                int x;
                cin>>x;
                string str;
                if(x!=-1 ){
                    cin>>str;
                }else if(x==-1){
                   cout<<st1.top().second<<" "<<st1.top().first.second<<endl;
                    st1.pop();
                    continue;
                }
                if (x==0){
                    continue;
                }
                if(!st1.empty() and st1.top().first.first<x ){
                   st1.top().second++;
                   continue;
                 }

                    pair<int,string> p = make_pair(x,str);
                    pair<pair<int,string>,int> whole_pair= make_pair(p,0);
                    st1.push(whole_pair);
            }
            return 0;
}
```



32. Longest Valid Parentheses

```
class Solution {
public:
    int longestValidParentheses(string s) {
        stack<int> stack;
        int n = s.size(), ans=0;
        stack.push(-1);

        for(int i=0;i<n;i++){
            if(s[i]=='('){
                stack.push(i);
            }
            else{
                stack.pop();
                if(stack.size()==0) stack.push(i);
                else{
                    ans = max(ans, i-stack.top());
                }
            }
        }
        return ans;
    }
};
```

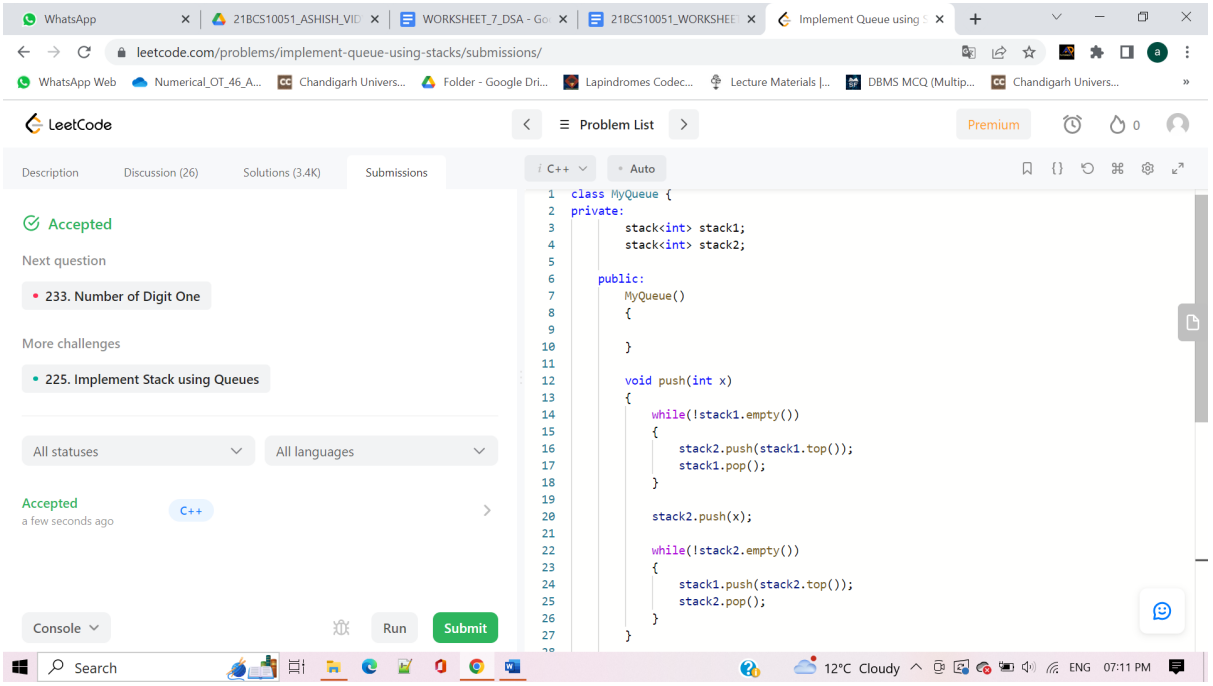


Min Stack

```
class MinStack
{

    Stack<Long> stack = new Stack<>();
    Long mini;

    public MinStack()
    {
        mini = Long.MAX_VALUE;
    }

    public void push(int val)
    {
        Long value = Long.valueOf(val);
        if(stack.isEmpty())
        {
            mini =  value;
            stack.push(value);
        }
        else
        {
            if(value < mini)
            {
                stack.push(2 * value - mini);
                mini = value;
```

```
            }
            else
            {
                stack.push(value);
            }
        }
    }

    public void pop()
    {
        if(stack.isEmpty())
        {
            return;
        }

        Long value = stack.pop();
        if(value < mini)
        {
            mini = 2 * mini - value;
        }

    }

    public int top()
    {
        Long value = stack.peek();
        if(value < mini)
        {
            return mini.intValue();
        }
        return value.intValue();
    }

    public int getMin()
    {
        return mini.intValue();
    }
}
```

. Implement Queue using Stacks



First Unique Character in a String