1.[Coin Change Problem](https://practice.geeksforgeeks.org/problems/coin-change2448/1)

int coinChange(vector<int>& coins, int amount) {

vector<int> s(amount + 1, amount + 1);

s[0] = 0;

for (int i = 1; i <= amount; i++) {

for (int j = 0; j < coins.size(); j++) {

if (i >= coins[j]) {

s[i] = min(s[i - coins[j]] + 1, s[i]);

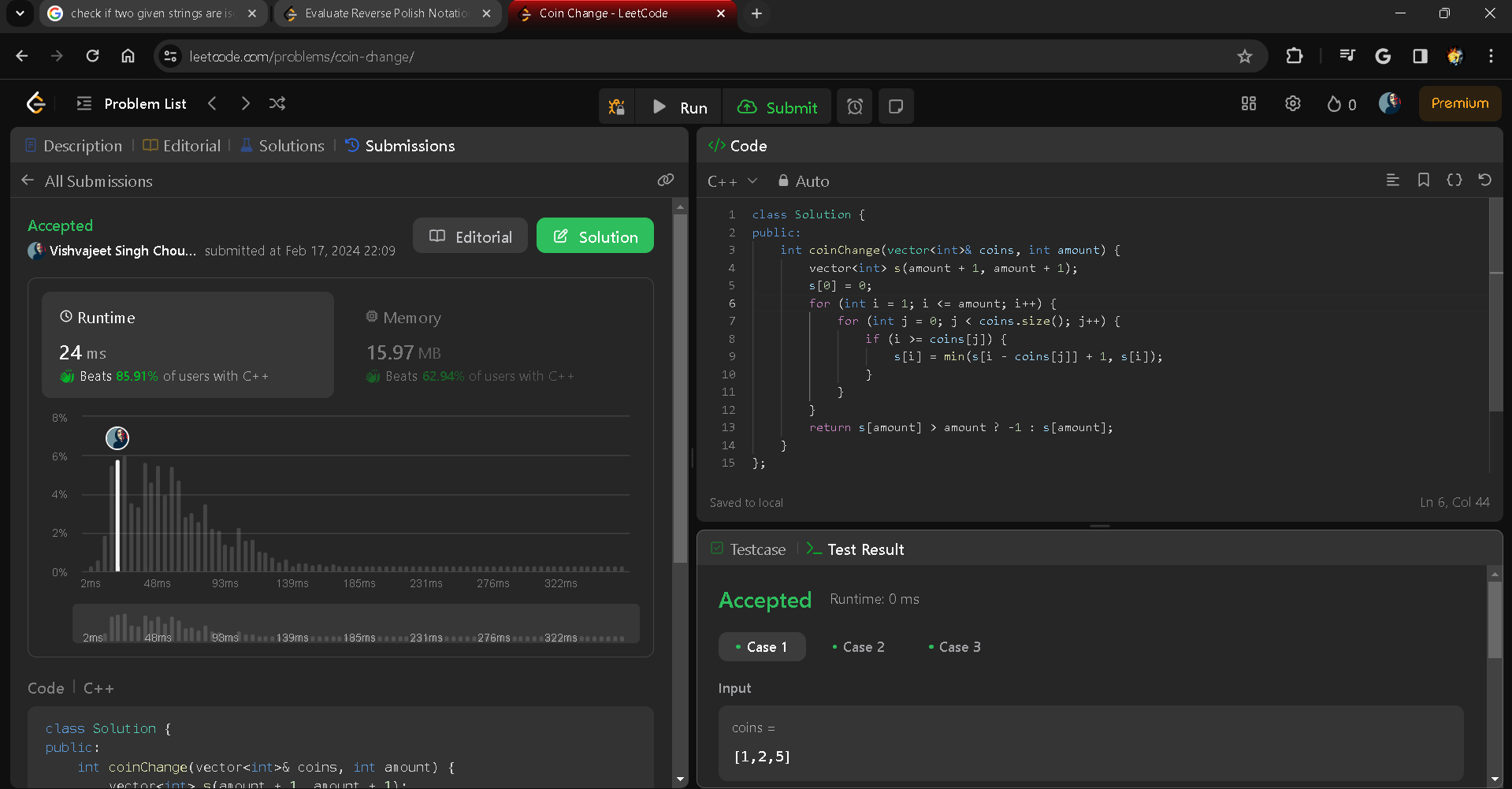
}

}

}

return s[amount] > amount ? -1 : s[amount];

}



Q2 [Knapsack Problem](https://practice.geeksforgeeks.org/problems/0-1-knapsack-problem/0)

int solver(int ind,int W, int wt[], int val[],vector<vector<int>> &dp){

if(ind==0){

if(W>=wt[ind]){

return val[ind];

}

return 0;

}

if(dp[ind][W]!=-1){

return dp[ind][W];

}

int take = INT\_MIN;

if(W>=wt[ind]){

take = val[ind] + solver(ind-1,W-wt[ind],wt,val,dp);

}

int nottake = solver(ind-1,W,wt,val,dp);

return dp[ind][W] = max(nottake,take);

}

int knapSack(int W, int wt[], int val[], int n)

{

vector<vector<int>> dp(n+1,vector<int> (W+1,-1));

return solver(n-1,W,wt,val,dp);

// Your code here

}

A screenshot of a computer program

Description automatically generated

Q3 [Edit Distance](https://practice.geeksforgeeks.org/problems/edit-distance3702/1)

int solver(string s, string t,int i,int j,vector<vector<int>> &dp){

// int n = s.length();

// int m = t.length();

if (i == -1) {

return j + 1;

}

if (j == -1) {

return i + 1;

}

if(dp[i][j]!=-1){

return dp[i][j];

}

if(s[i]==t[j]){

return dp[i][j] = solver(s,t,i-1,j-1,dp);

}

else{

int l = 0;

int r = 0;

int p = 0;

l = 1 + solver(s,t,i,j-1,dp);

r = 1 + solver(s,t,i-1,j,dp);

p = 1 + solver(s,t,i-1,j-1,dp);

return dp[i][j] = min(min(l,r),p);

}

}

int editDistance(string s, string t) {

int n = s.size();

int m = t.size();

vector<vector<int>> dp(n,vector<int> (m,-1));

return solver(s,t,n-1,m-1,dp);

// Code here

}

A screenshot of a computer

Description automatically generated

Q4 [Subset Sum Problem](https://practice.geeksforgeeks.org/problems/subset-sum-problem2014/1)

bool canPartition(vector<int>& nums) {

int totalSum = accumulate(begin(nums), end(nums), 0);

if (totalSum % 2 != 0) {

return false;

}

int halfSum = totalSum / 2;

vector<bool> dp(halfSum + 1, false);

dp[0] = true;

for (int i = 0; i < nums.size(); i++) {

for (int j = halfSum; j >= nums[i]; j--) {

if (dp[j - nums[i]])

dp[j] = true;

}

}

return dp[halfSum]; }

A screenshot of a computer

Description automatically generated

Q5 [Longest Common Subsequence](https://practice.geeksforgeeks.org/problems/longest-common-subsequence/0)

int memo\_helper(int i , int j , string s1 , string s2 , vector<vector<int>> dp){

if(i == 0 || j == 0) return 0;

if(dp[i][j] != -1) return dp[i][j];

if(s1[i - 1] == s2[j - 1])

return dp[i][j] = 1 + memo\_helper(i - 1 , j - 1 , s1 , s2 , dp);

return dp[i][j] = max(memo\_helper(i, j - 1 , s1 , s2 , dp) , memo\_helper(i - 1, j, s1 , s2 , dp));

}

int memo(int n , int m , string s1 , string s2){

vector<vector<int>> dp(n + 1 , vector<int>(m + 1 , -1));

// shifting the index for tabulation

return memo\_helper(n , m, s1 , s2 , dp);

}

int longestCommonSubsequence(string s1, string s2) {

int n = s1.size();

int m = s2.size();

int ans = memo(n, m , s1 , s2);

return ans;

}

A screenshot of a computer

Description automatically generated

Q6 [Longest Increasing Subsequence](https://practice.geeksforgeeks.org/problems/longest-increasing-subsequence/0)

int lengthOfLIS(std::vector<int>& nums) {

if (nums.empty()) {

return 0;

}

int n = nums.size();

std::vector<int> dp(n, 1);

for (int i = 1; i < n; ++i) {

for (int j = 0; j < i; ++j) {

if (nums[i] > nums[j]) {

dp[i] = std::max(dp[i], dp[j] + 1);

}

}

}

return \*std::max\_element(dp.begin(), dp.end());

}

A screenshot of a computer

Description automatically generated

Q7 [Egg Dropping Problem](https://practice.geeksforgeeks.org/problems/egg-dropping-puzzle/0)

int helper(int k, int n, vector<vector<int>> &dp){

if(n==0 or n==1 or k==1) return n;

if(dp[n][k]!=-1) return dp[n][k];

int low=1, high=n, result=high;

while(low<=high){

int mid=low+(high-low)/2;

int lower=0, upper=0;

if(dp[mid-1][k-1]!=-1){

lower=dp[mid-1][k-1];

}else{

lower=helper(k-1,mid-1,dp);

dp[mid-1][k-1]=lower;

}

if(dp[n-mid][k]!=-1){

upper=dp[n-mid][k];

}else{

upper=helper(k,n-mid,dp);

dp[n-mid][k]=upper;

}

int temp=1+max(lower,upper);

dp[n][k]=temp;

result=min(result,temp);

if(lower<upper) low=mid+1;

else if(lower>upper) high=mid-1;

else break;

}

return dp[n][k]=result;

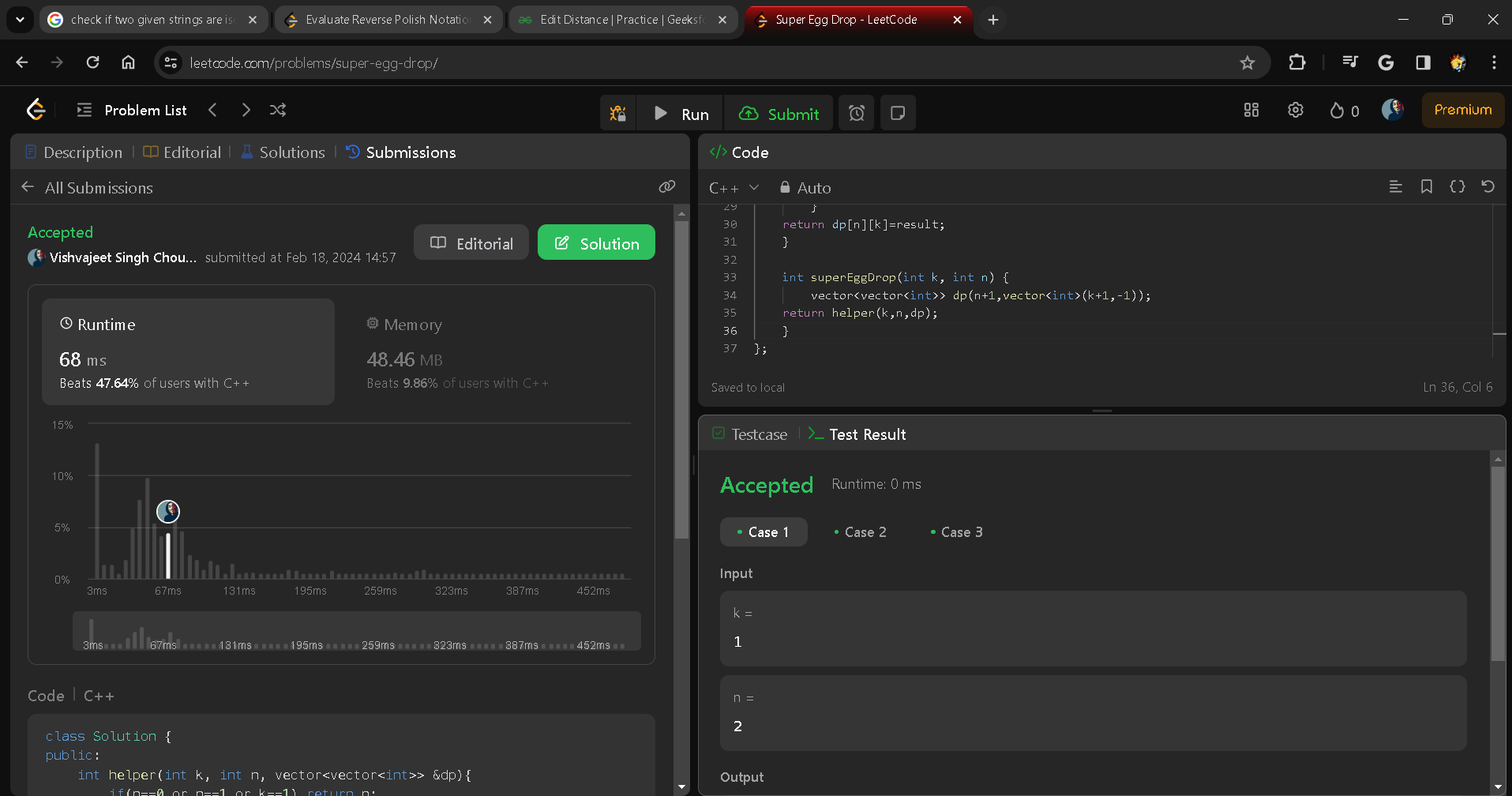
}

int superEggDrop(int k, int n) {

vector<vector<int>> dp(n+1,vector<int>(k+1,-1));

return helper(k,n,dp);

}



Q8 [Unbounded Knapsack (Repetition of items allowed)](https://practice.geeksforgeeks.org/problems/knapsack-with-duplicate-items4201/1)

int coinChange(vector<int>& coins, int amount) {

vector<int> s(amount + 1, amount + 1);

s[0] = 0;

for (int i = 1; i <= amount; i++) {

for (int j = 0; j < coins.size(); j++) {

if (i >= coins[j]) {

s[i] = min(s[i - coins[j]] + 1, s[i]);

}

}

}

return s[amount] > amount ? -1 : s[amount];

}

A screenshot of a computer

Description automatically generated

Q9 [Word Break Problem](https://practice.geeksforgeeks.org/problems/word-break/0)

bool wordBreak(string s, vector<string>& wordDict) {

const int n = s.length();

const int maxLength = getMaxLength(wordDict);

const unordered\_set<string> wordSet{begin(wordDict), end(wordDict)};

vector<int> dp(n + 1);

dp[0] = true;

for (int i = 1; i <= n; ++i)

for (int j = i - 1; j >= 0; --j) {

if (i - j > maxLength)

break;

if (dp[j] && wordSet.count(s.substr(j, i - j))) {

dp[i] = true;

break;

}

}

return dp[n];

}

private:

int getMaxLength(const vector<string>& wordDict) {

return max\_element(begin(wordDict), end(wordDict),

[](const auto& a, const auto& b) {

return a.length() < b.length();

})

->length();

}

A screenshot of a computer

Description automatically generated

Q10 [Longest Palindromic Subsequence](https://www.geeksforgeeks.org/longest-palindromic-subsequence-dp-12/)

int n=A.length();

string B=A;

reverse(A.begin(),A.end());

vector<vector<int>>dp(n+1,vector<int>(n+1,0));

for(int i=1;i<n+1;i++)

{

for(int j=1;j<n+1;j++)

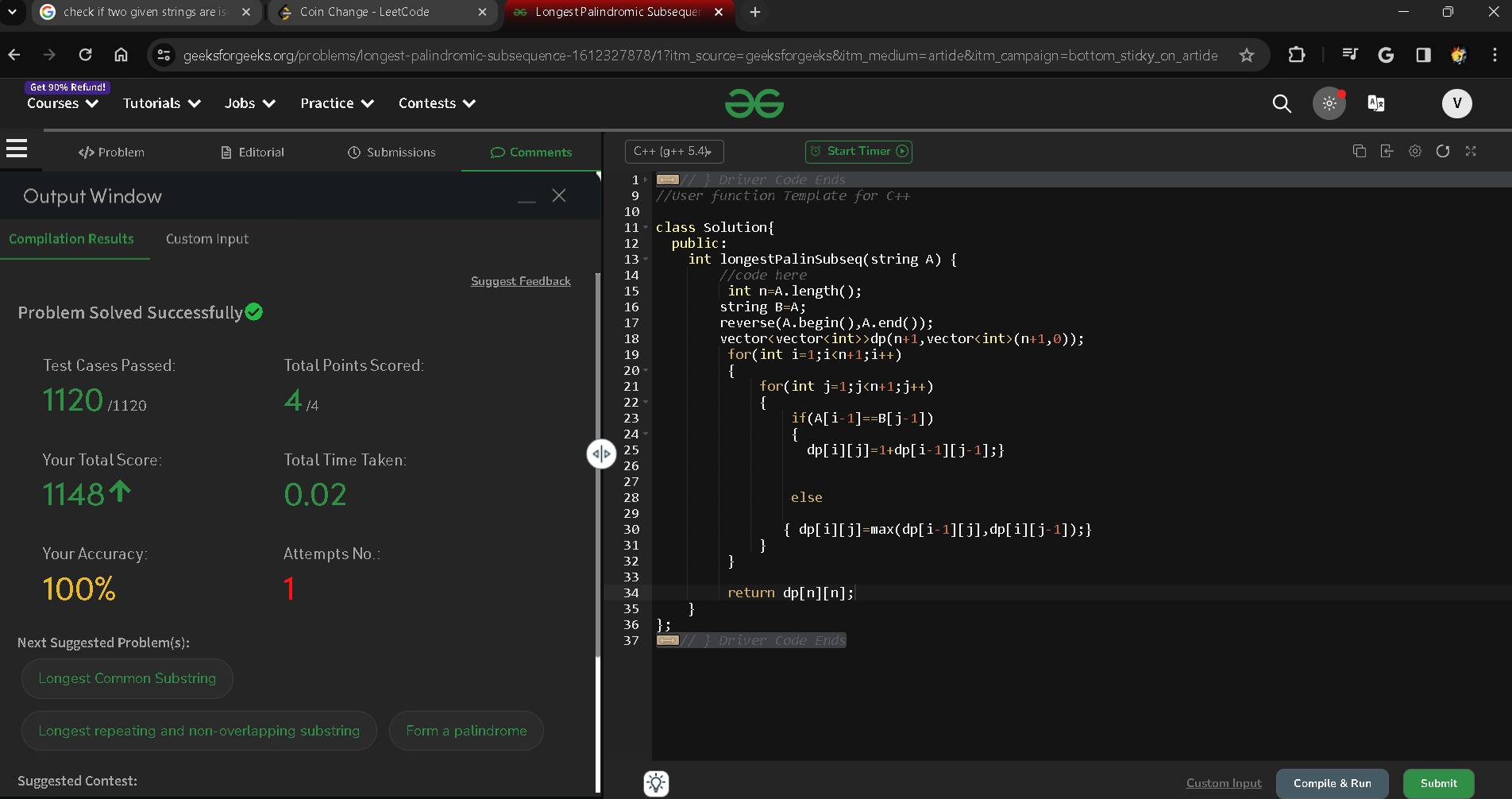
{

if(A[i-1]==B[j-1])

{

dp[i][j]=1+dp[i-1][j-1];}

else { dp[i][j]=max(dp[i-1][j],dp[i][j-1]);} } } return dp[n][n];



Q11 [Weighted Job Scheduling](https://www.geeksforgeeks.org/weighted-job-scheduling/)

int jobScheduling(vector<int>& startTime, vector<int>& endTime, vector<int>& profit) {

int numJobs = profit.size(); // Number of jobs

vector<tuple<int, int, int>> jobs(numJobs);

for (int i = 0; i < numJobs; ++i) {

jobs[i] = {endTime[i], startTime[i], profit[i]};

}

sort(jobs.begin(), jobs.end());

vector<int> dp(numJobs + 1);

for (int i = 0; i < numJobs; ++i) {

auto [endTime, startTime, profit] = jobs[i];

int latestNonConflictJobIndex = upper\_bound(jobs.begin(), jobs.begin() + i, startTime, [&](int time, const auto& job) -> bool {

return time < get<0>(job);

}) - jobs.begin();

dp[i + 1] = max(dp[i], dp[latestNonConflictJobIndex] + profit);

}

return dp[numJobs];

}

A screenshot of a computer

Description automatically generated