**NAME: MUKUL KUMAR UID: 22BET10306**

**Problem 1**

**Aim:**

Longest Increasing Subsequence

**Code:**

class Solution {

public:

int lengthOfLIS(vector<int>& nums) {

vector<int> res;

for (int n : nums) {

if (res.empty() || res.back() < n) {

res.push\_back(n);

} else {

int idx = binarySearch(res, n);

res[idx] = n;

}

return res.size();

}private:

int binarySearch(const vector<int>& arr, int target) {

int left = 0;

int right = arr.size() - 1;

while (left <= right) {

int mid = (left + right) / 2;

if (arr[mid] == target) {

return mid;

} else if (arr[mid] > target) {

right = mid - 1;

} else {

left = mid + 1;

}

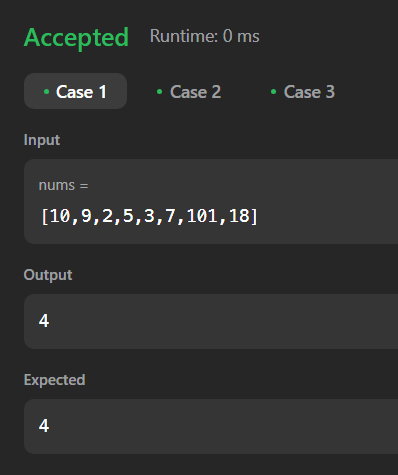
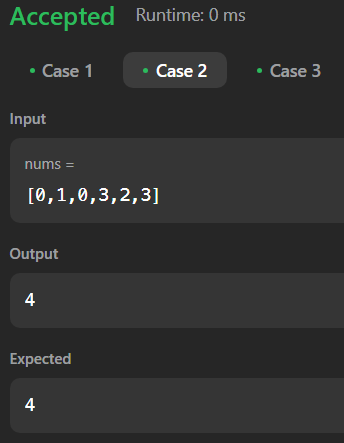
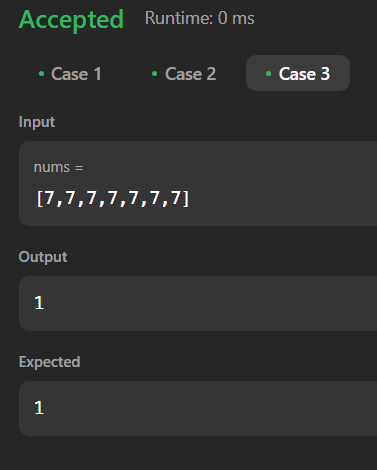
}

return left;

}

};

**Output:**

Case 1 Case 2 Case 3

**Problem 2**

**Aim:**

Maximum Product Subarray

**Code:**

class Solution {

public:

int maxProduct(vector<int>& nums) {

int maxi = INT\_MIN;

int prod=1;

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

{

prod\*=nums[i];

maxi=max(prod,maxi);

if(prod==0)

prod=1;

}

prod=1;

for(int i=nums.size()-1;i>=0;i--)

{

prod\*=nums[i];

maxi=max(prod,maxi);

if(prod==0)

prod=1;

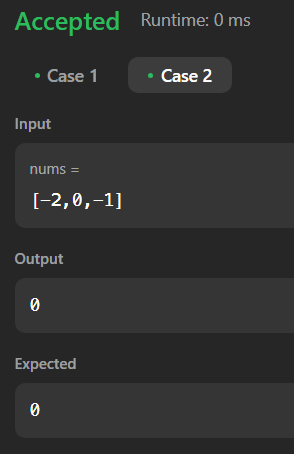
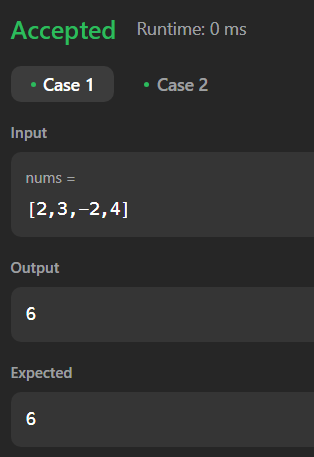
}

return maxi;

}

};

**Output:**



Test Case 1 Test Case 2

**Problem 3**

**Aim:**

Decode Ways

**Code:**

class Solution {

public:

int numDecodings(string s) {

if (s[0] == '0') {

return 0;

}

int n = s.length();

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

dp[0] = dp[1] = 1;

for (int i = 2; i <= n; i++) {

int one = s[i - 1] - '0';

int two = stoi(s.substr(i - 2, 2));

if (1 <= one && one <= 9) {

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

}

if (10 <= two && two <= 26) {

dp[i] += dp[i - 2];

}

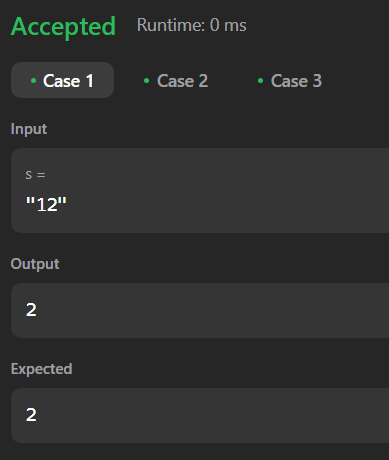
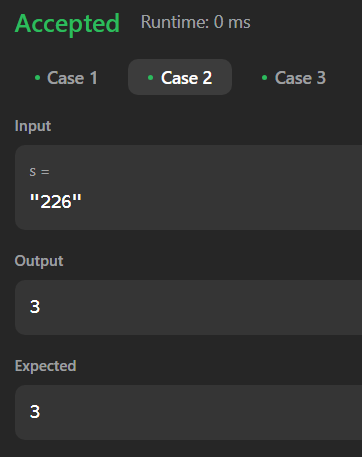
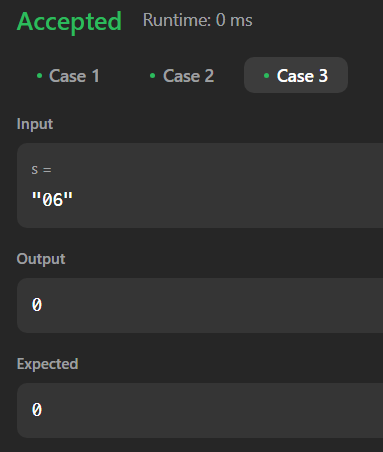
}

return dp[n];

}

};

**Output:**

**  **

Case 1 Case 2 Case 3

**Problem 4**

**Aim:**

Perfect Squares

**Code:**

class Solution {

public:

int numSquares(int n) {

vector<int> dp(n + 1, INT\_MAX);

dp[0] = 0;

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

for (int j = 1; j \* j <= i; ++j){

dp[i] = min(dp[i], dp[i - j \* j] + 1);

}

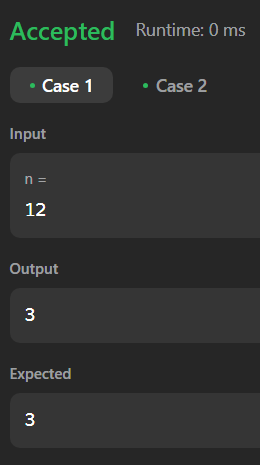
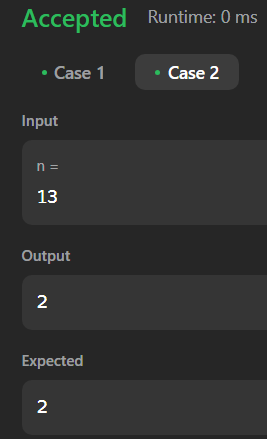
}

return dp[n];

}

};

**Output:**

** **

Case 1 Case 2

**Problem 5**

**Aim:**

Word Break

**Code:**

class Solution {

public:

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

vector<bool> dp(s.size() + 1, false);

dp[0] = true;

for (int i = 1; i <= s.size(); i++) {

for (const string& w : wordDict) {

int start = i - w.length();

if (start >= 0 && dp[start] && s.substr(start, w.length()) == w) {

dp[i] = true;

break;

}

}

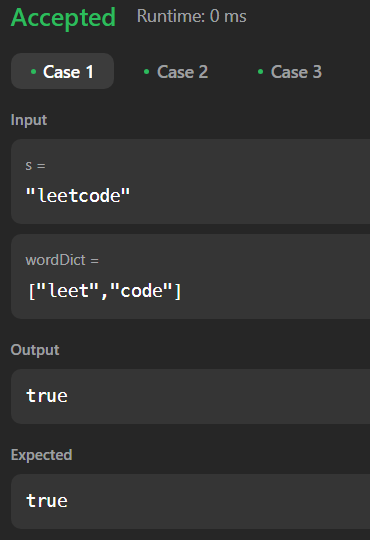
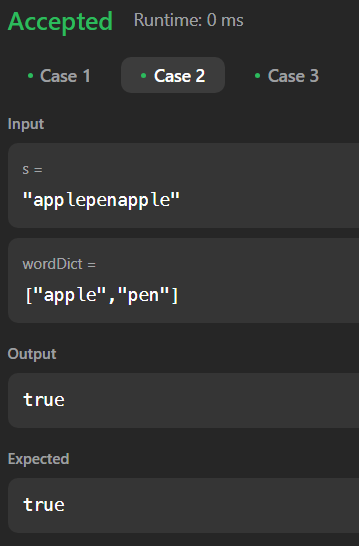
}

return dp[s.size()];

}

};

**Output:**

**  A screenshot of a computer

Description automatically generated**

Case 1 Case 2 Case 3

**Problem 6**

**Aim:**

Word Break 2

**Code:**

class Solution {

public:

void solve(string s, vector<string>& res, unordered\_set<string>& st, vector<string>&temp){

if(s.length() == 0){

string str = "";

for(auto it:temp){

str += it + " ";

}

str.pop\_back();

res.push\_back(str);

return;

}

for(int i=0;i<s.length(); i++){

if(st.count(s.substr(0, i+1))){

temp.push\_back(s.substr(0, i+1));

solve(s.substr(i+1), res, st, temp);

temp.pop\_back();

}

}

}

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

vector<string>res, temp;

unordered\_set<string>st(wordDict.begin(), wordDict.end());

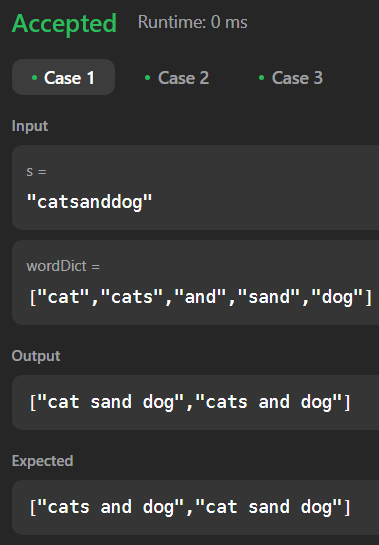
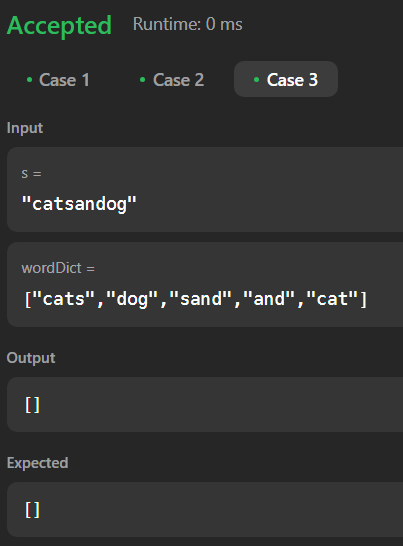
solve(s, res, st, temp);

return res;

}

};

**Output:**

Case 1 Case 3

A screenshot of a computer

Description automatically generated

Case 2

**Problem 7**

**Aim:**

Best time to buy and Sell a Stock with Cooldown

**Code:**

class Solution {

public int maxProfit(int[] prices) {

if (prices == null || prices.length <= 1) return 0;

int b0 = -prices[0], b1 = b0;

int s0 = 0, s1 = 0, s2 = 0;

for (int i = 1; i < prices.length; i++) {

b0 = Math.max(b1, s2 - prices[i]);

s0 = Math.max(s1, b1 + prices[i]);

b1 = b0;

s2 = s1;

s1 = s0;

}

return s0;

}

public static void main(String[] args) {

Solution solution = new Solution();

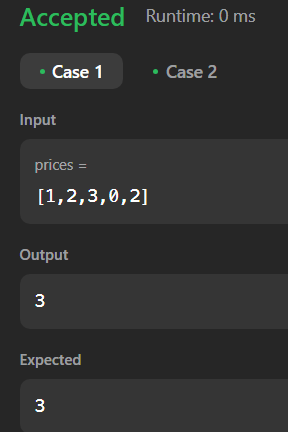
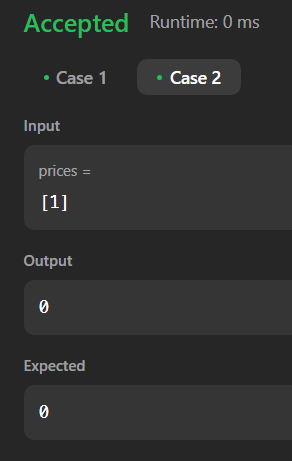
int[] prices = {1, 2, 3, 0, 2}; // Example input

System.out.println("Max Profit: " + solution.maxProfit(prices));

}

}

**Output:**

** **

Case 1 Case 2