**MILESTONE - 1**

**CP BASIC**

1. Jewels and Stones

class Solution {

public:

int numJewelsInStones(string j ,string s) {

int n = j.size(),m=s.size();

int i,count=0;

map<char,bool> mp;

for(i=0;i<n;i++)

{

mp[j[i]] = true;

}

for(i=0;i<m;i++)

{

if(mp[s[i]])

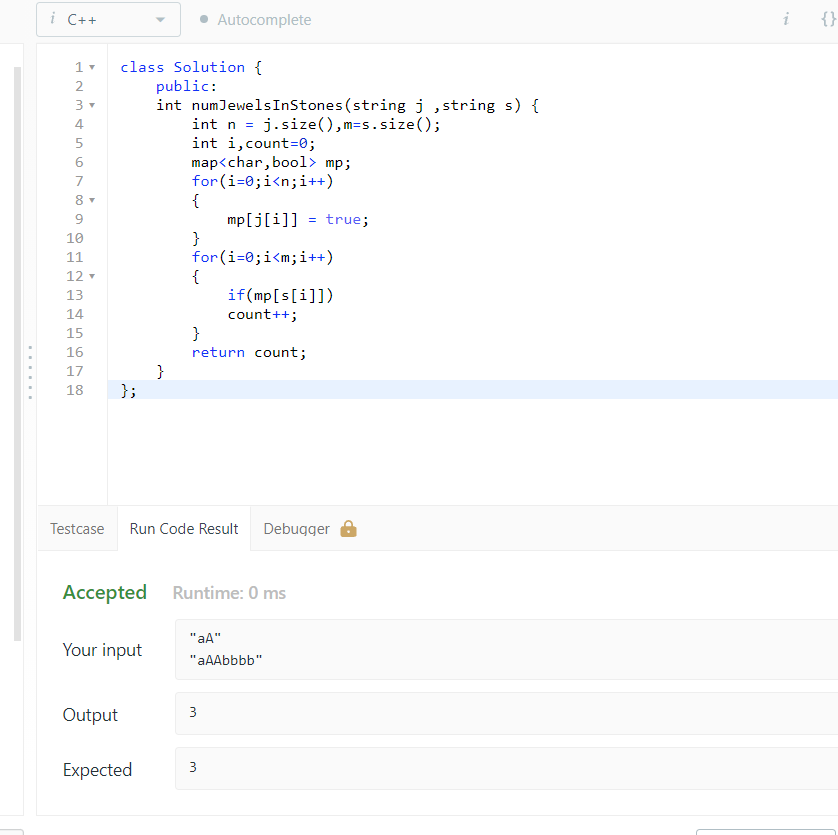
count++;

}

return count;

}

};



1. Merge string alternatively

class Solution {

public:

string mergeAlternately(string word1, string word2) {

string result;

for (int i = 0; i < size(word1) || i < size(word2); ++i) {

if (i < size(word1)) {

result.push\_back(word1[i]);

}

if (i < size(word2)) {

result.push\_back(word2[i]);

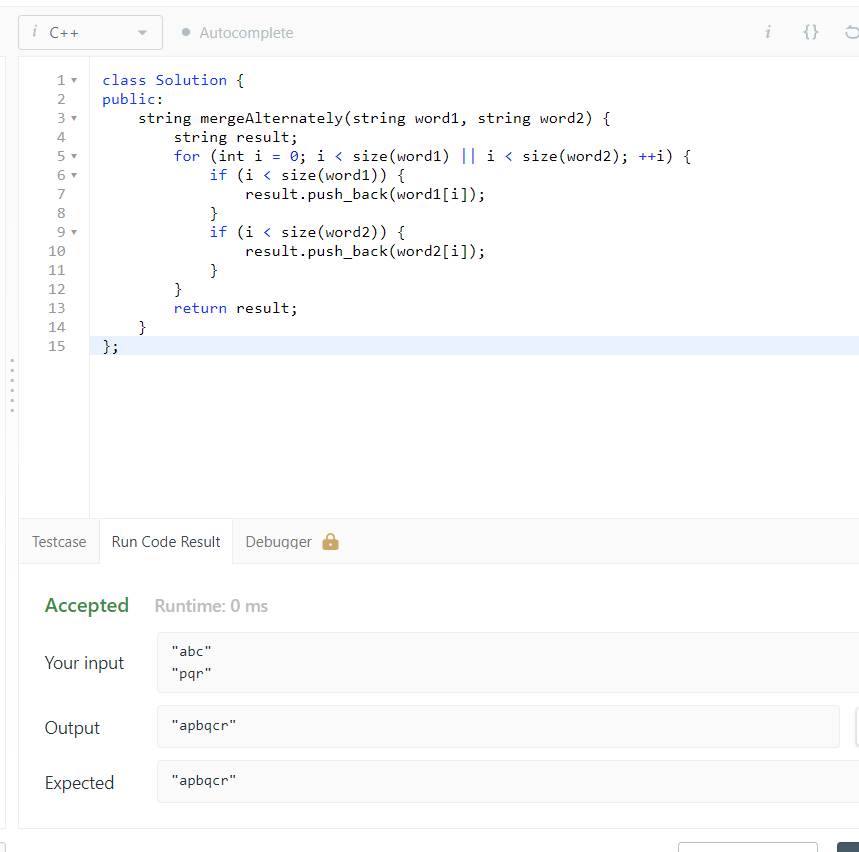
}

}

return result;

}

};



1. Minimum number of steps to make two strings anagram

class Solution

{

public:

int minSteps(string s, string t)

{

int arr1[26] = {0};

int arr2[26] = {0};

int n = s.size();

for (int i = 0; i < n; ++i)

{

++arr1[s[i] - 'a'];

++arr2[t[i] - 'a'];

}

int ans = 0;

for (int i = 0; i < 26; ++i)

{

if (arr1[i] > arr2[i])

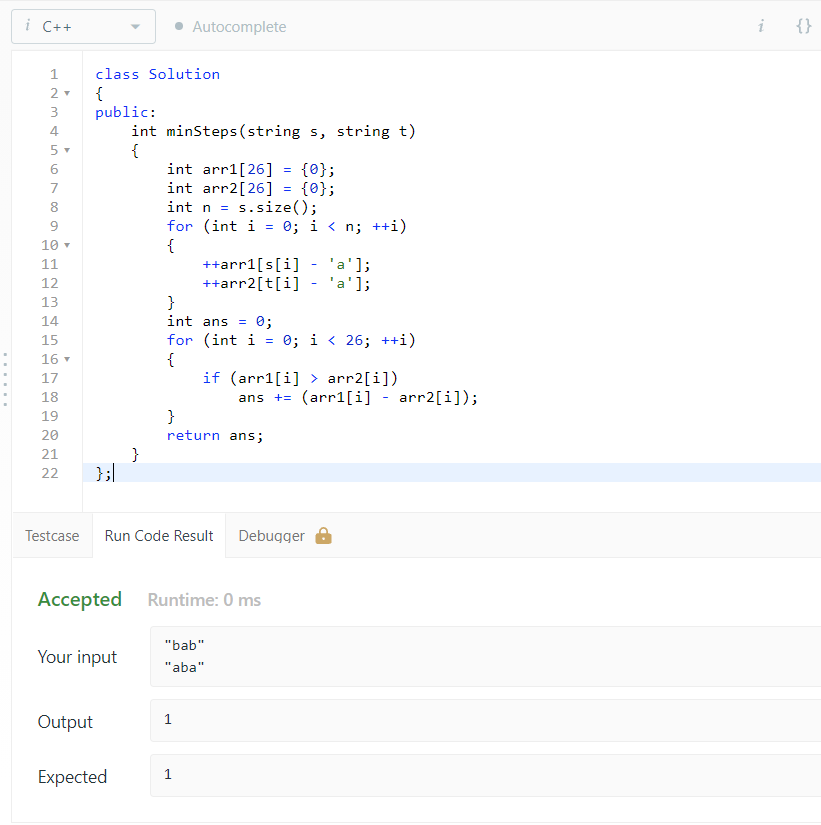
ans += (arr1[i] - arr2[i]);

}

return ans;

}

};



1. Spiral Matrix

class Solution {

public:

vector<int> spiralOrder(vector<vector<int>>& matrix) {

int m = matrix.size(), n = m ? matrix[0].size() : 0, u = 0, d = m - 1, l = 0, r = n - 1, p = 0;

vector<int> order(m \* n);

while (u <= d && l <= r) {

for (int col = l; col <= r; col++) {

order[p++] = matrix[u][col];

}

if (++u > d) {

break;

}

for (int row = u; row <= d; row++) {

order[p++] = matrix[row][r];

}

if (--r < l) {

break;

}

for (int col = r; col >= l; col--) {

order[p++] = matrix[d][col];

}

if (--d < u) {

break;

}

for (int row = d; row >= u; row--) {

order[p++] = matrix[row][l];

}

if (l++ > r) {

break;

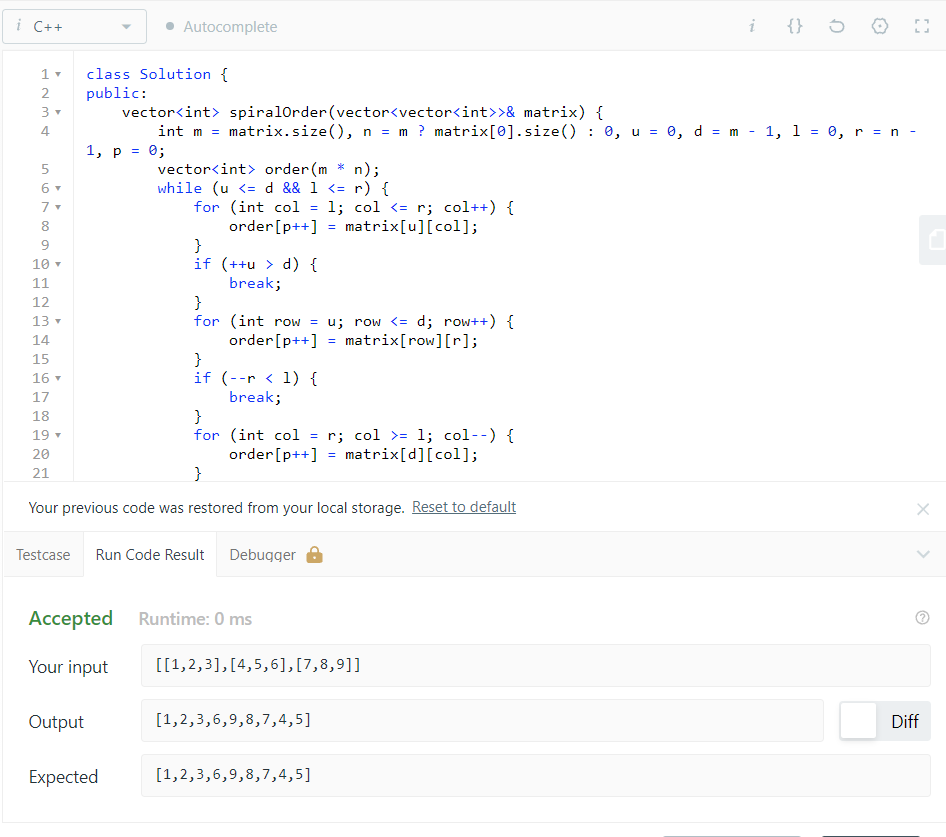
}

}

return order;

}

};



1. Sort array by parity

class Solution {

public:

vector<int> sortArrayByParity(vector<int>& A) {

// i for vefify, j for traverse

for(int i=0, j=0; j < A.size(); j++){

if( A[j] % 2 == 0 ){

swap(A[i], A[j]);

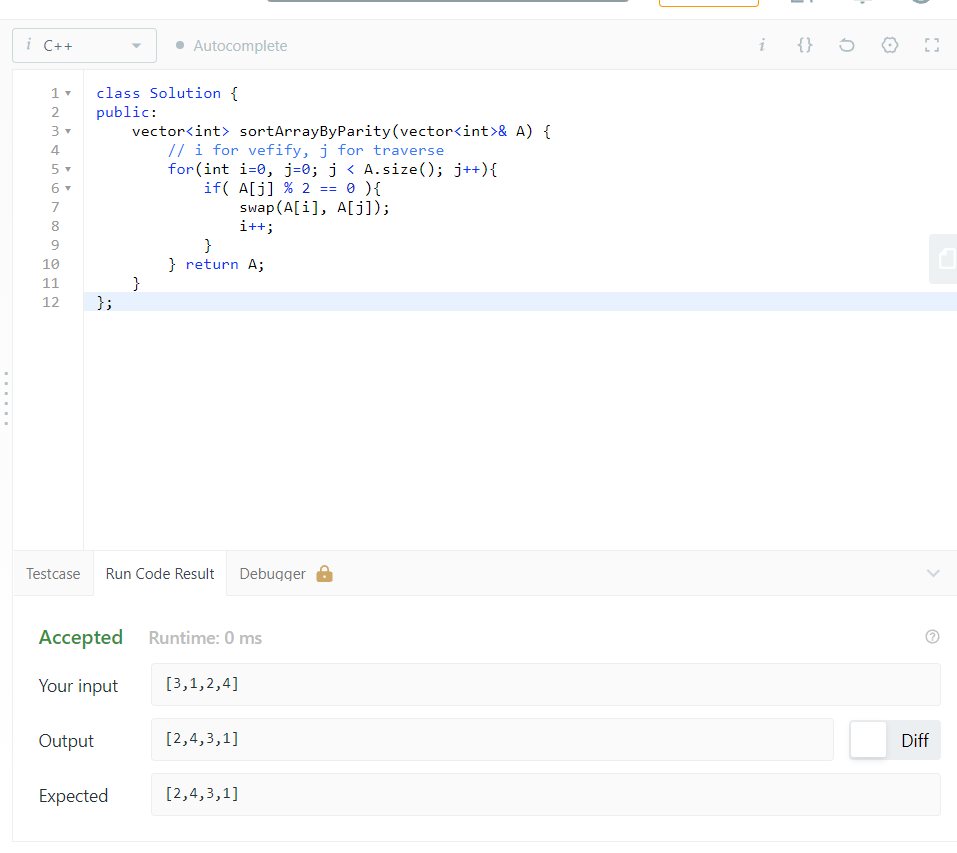
i++;

}

} return A;

}

};



1. Best time to buy and sell stock

class Solution {

public:

int maxProfit(vector<int>& prices) {

if(prices.size() == 0) return 0;

int ans = 0;

int start = prices[0], end = prices[0];

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

if(prices[i] < start){

//restart as session

ans = max(ans, end - start);

start = prices[i];

end = prices[i];

}else{

//continue current session

end = max(end, prices[i]);

}

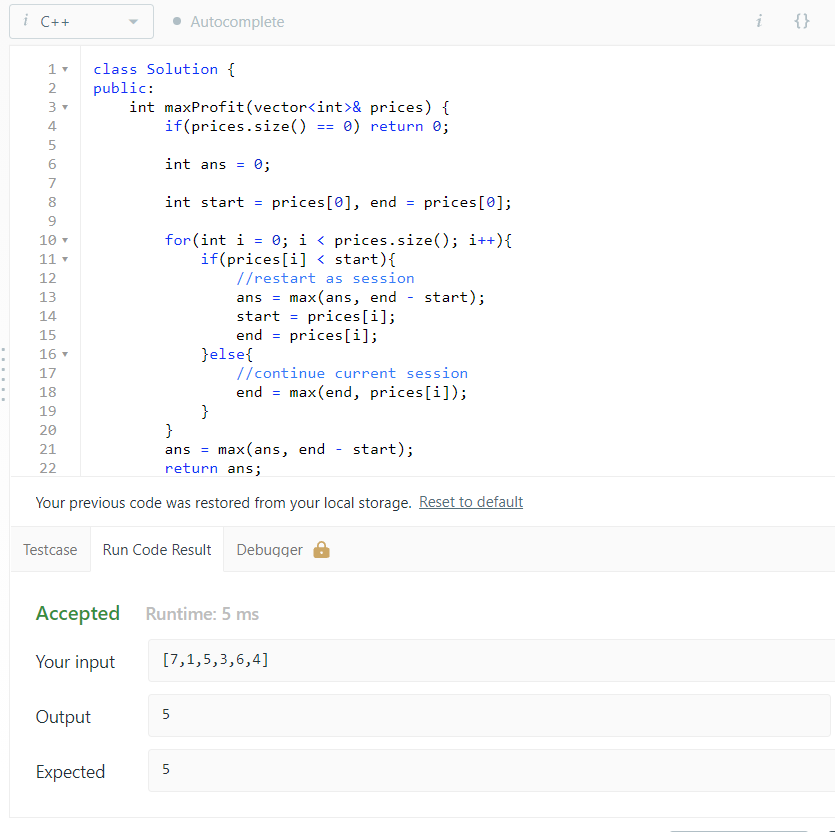
}

ans = max(ans, end - start);

return ans;

}

};



1. Best time to buy and sell stock -ii

class Solution {

public:

int maxProfit(vector<int>& prices) {

int mp = 0;

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

{

if(prices[i] > prices[i-1])

{

mp += prices[i]-prices[i-1];

}

}

return mp;

}

};



LeetCode id: **https://leetcode.com/kashish\_1129/**