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;
    }
};
```

C++ Autocomplete

```cpp
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;
    }
};
```

Testcase | Run Code Result | Debugger

Accepted Runtime: 0 ms

Your input	"aA" "aAAbbbb"
Output	3
Expected	3

2. 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;
    }
};
```

C++ Autocomplete

```cpp
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;
    }
};
```

Testcase | Run Code Result | Debugger

Accepted Runtime: 0 ms

Your input	"abc" "pqr"
Output	"apbqcr"
Expected	"apbqcr"

3. 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;
    }
};
```

C++ Autocomplete

```cpp
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;
    }
};
```

Testcase | Run Code Result | Debugger

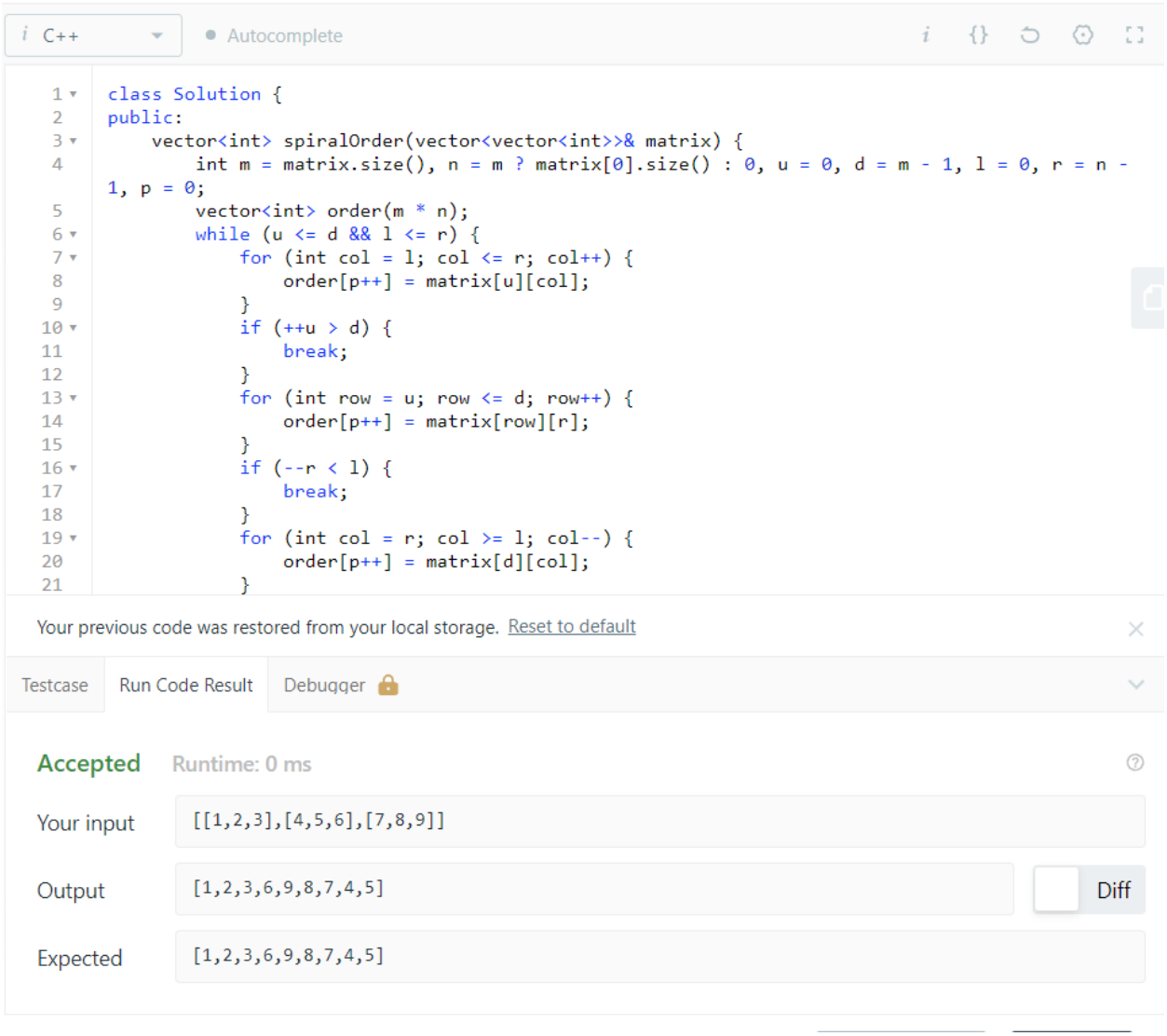
Accepted Runtime: 0 ms

Your input	"bab" "aba"
Output	1
Expected	1

4. Spiral Matrix

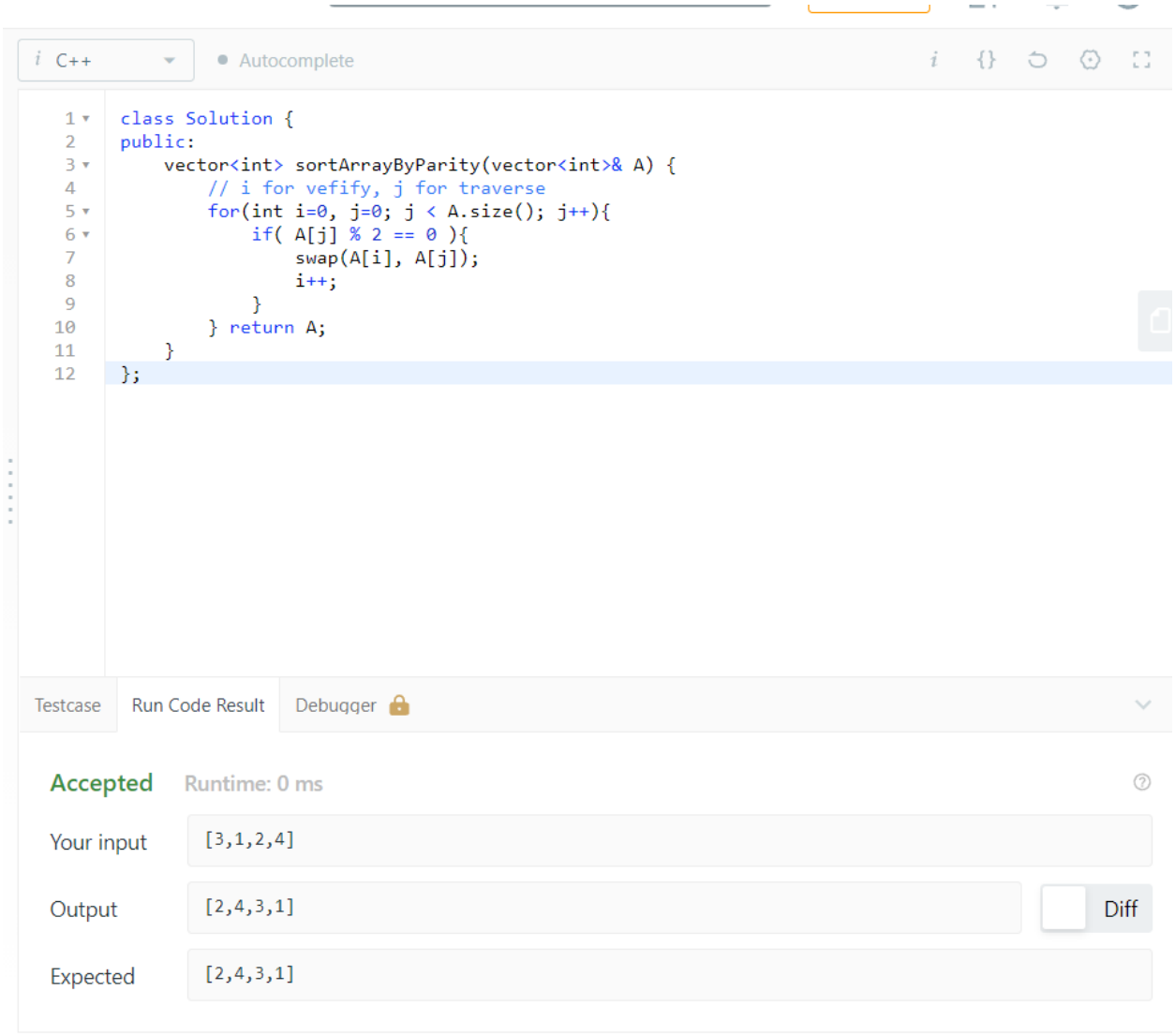
```
class Solution {
public:
    vector<int> spiralOrder(vector<vector<int>>& matrix) {
        int m = matrix.size(), n = m ? matrix[0].size() : 0, u = 0, d = m - 1, l = 0,
r = n - 1, p = 0;
        vector<int> order(m * n);
        while (u <= d && l <= r) {
            for (int col = l; col <= r; 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;
    }
};
```



5. 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;
    }
};
```



6. Best time to buy and sell stock

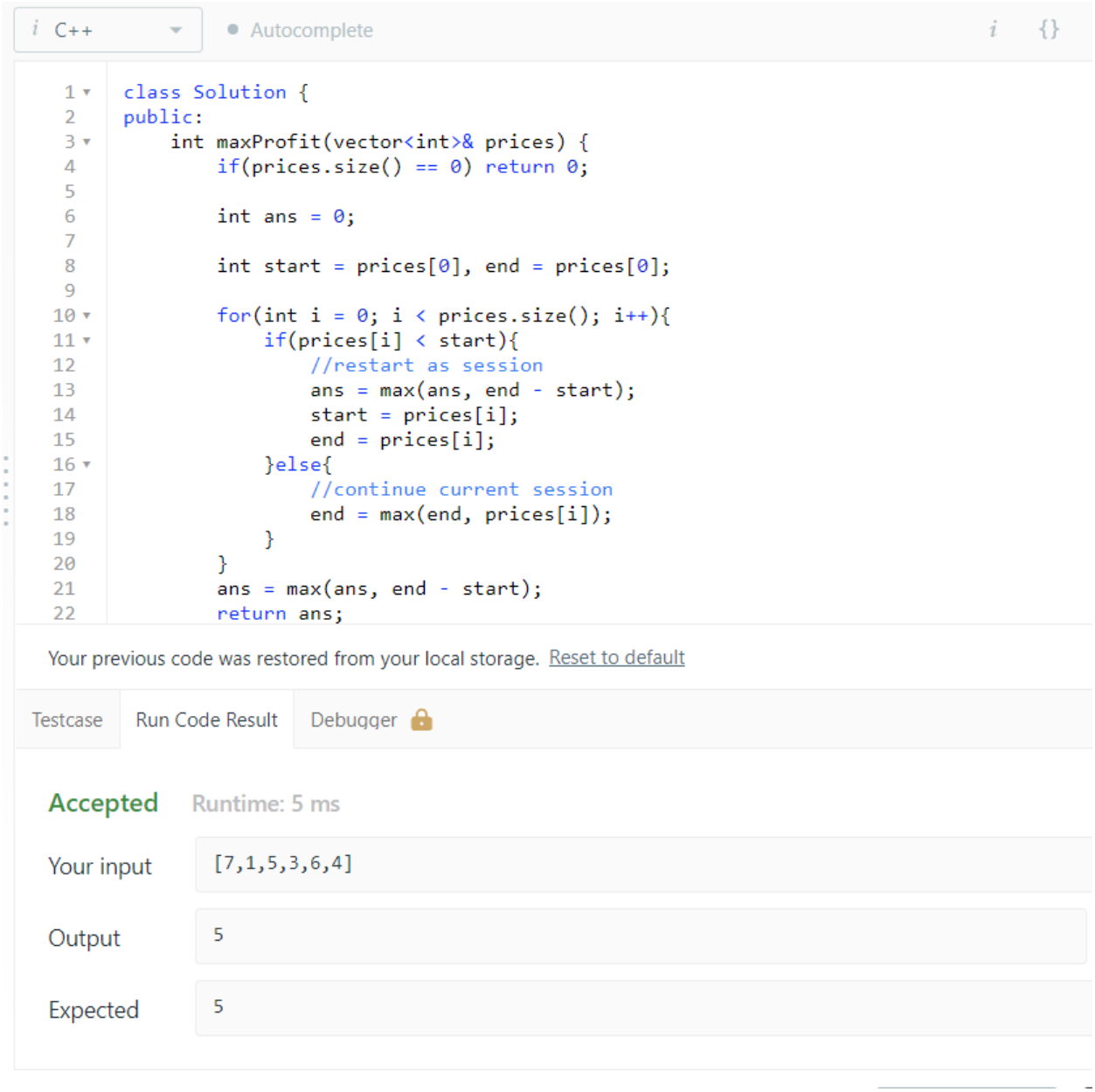
```
class Solution {
public:
    int maxProfit(vector<int>& prices) {
        if(prices.size() == 0) return 0;

        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;
    }
};
```



7. 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;
    }
};
```

C++ Autocomplete

```cpp
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;
    }
};
```

Testcase | Run Code Result | Debugger

Accepted Runtime: 3 ms

Your input	[7,1,5,3,6,4]
Output	7
Expected	7

LeetCode id: **https://leetcode.com/kashish_1129/**