

# 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;
    }
};

```

i C++


Autocomplete

i {}

```
1 class Solution {
2     public:
3     int numJewelsInStones(string j ,string s) {
4         int n = j.size(),m=s.size();
5         int i,count=0;
6         map<char,bool> mp;
7         for(i=0;i<n;i++)
8         {
9             mp[j[i]] = true;
10        }
11        for(i=0;i<m;i++)
12        {
13            if(mp[s[i]])
14                count++;
15        }
16        return count;
17    }
18 };
```

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;
}
};
```

i C++Autocomplete

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

```
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;
    }
};
```

i C++


Autocomplete

i {}

```
1  class Solution
2  {
3  public:
4      int minSteps(string s, string t)
5      {
6          int arr1[26] = {0};
7          int arr2[26] = {0};
8          int n = s.size();
9          for (int i = 0; i < n; ++i)
10         {
11             ++arr1[s[i] - 'a'];
12             ++arr2[t[i] - 'a'];
13         }
14         int ans = 0;
15         for (int i = 0; i < 26; ++i)
16         {
17             if (arr1[i] > arr2[i])
18                 ans += (arr1[i] - arr2[i]);
19         }
20         return ans;
21     }
22 };
```

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;
}
};

```

C++
Autocomplete
i {} ↺ ⚙️ 🗑️

```

1 class Solution {
2 public:
3     vector<int> spiralOrder(vector<vector<int>>& matrix) {
4         int m = matrix.size(), n = m ? matrix[0].size() : 0, u = 0, d = m - 1, l = 0, r = n -
1, p = 0;
5         vector<int> order(m * n);
6         while (u <= d && l <= r) {
7             for (int col = l; col <= r; col++) {
8                 order[p++] = matrix[u][col];
9             }
10            if (++u > d) {
11                break;
12            }
13            for (int row = u; row <= d; row++) {
14                order[p++] = matrix[row][r];
15            }
16            if (--r < l) {
17                break;
18            }
19            for (int col = r; col >= l; col--) {
20                order[p++] = matrix[d][col];
21            }

```

Your previous code was restored from your local storage. [Reset to default](#)

Testcase
Run Code Result
Debugger 🔒

Accepted Runtime: 0 ms ⓘ

Your input
[[1,2,3],[4,5,6],[7,8,9]]

Output
[1,2,3,6,9,8,7,4,5]
☐ Diff

Expected
[1,2,3,6,9,8,7,4,5]



## 5. Sort array by parity

```
class Solution {
public:
    vector<int> sortArrayByParity(vector<int>& A) {
        // i for verify, 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;
    }
};
```

```
i C++ Autocomplete i {} ↺ ⚙️ 📄
```

```
1 class Solution {  
2 public:  
3     vector<int> sortArrayByParity(vector<int>& A) {  
4         // i for verify, j for traverse  
5         for(int i=0, j=0; j < A.size(); j++){  
6             if( A[j] % 2 == 0 ){  
7                 swap(A[i], A[j]);  
8                 i++;  
9             }  
10        } return A;  
11    }  
12};
```

Testcase Run Code Result Debugger 🔒

**Accepted** Runtime: 0 ms ⓘ

Your input [3,1,2,4]

Output [2,4,3,1] ☐ Diff

Expected [2,4,3,1]

## 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;
}
};
```

*i* C++
 Autocomplete
 *i* {}

```

1 class Solution {
2 public:
3     int maxProfit(vector<int>& prices) {
4         if(prices.size() == 0) return 0;
5
6         int ans = 0;
7
8         int start = prices[0], end = prices[0];
9
10        for(int i = 0; i < prices.size(); i++){
11            if(prices[i] < start){
12                //restart as session
13                ans = max(ans, end - start);
14                start = prices[i];
15                end = prices[i];
16            }else{
17                //continue current session
18                end = max(end, prices[i]);
19            }
20        }
21        ans = max(ans, end - start);
22        return ans;
  
```

Your previous code was restored from your local storage. [Reset to default](#)

Testcase
 Run Code Result
 Debugger

**Accepted** Runtime: 5 ms

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

Output
 5

Expected
 5

## 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;
}
};
```

i C++Autocomplete

1class Solution {  
2public:  
3int maxProfit(vector<int>& prices) {  
4  
5int mp = 0;  
6  
7for(int i =1; i<prices.size(); i++)  
8{  
9if(prices[i] > prices[i-1])  
10{  
11mp += prices[i]-prices[i-1];  
12}  
13}  
14return mp;  
15}  
16};

TestcaseRun Code ResultDebugger

Accepted Runtime: 3 ms

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

Output 7

Expected 7

LeetCode id: [https://leetcode.com/kashish\\_1129/](https://leetcode.com/kashish_1129/)