```cpp
class Solution {
public:
    vector<int> twoSum(vector<int>& nums, int target) {
        int n = nums.size();
        for (int i = 0; i < n - 1; i++) {
            for (int j = i + 1; j < n; j++) {
                if (nums[i] + nums[j] == target) {
                    return {i, j};
                }
            }
        }
        return {}; // No solution found
    }
};
```

Saved

Ln 1, Col 1

Testcase | Test Result

Case 1 Case 2 Case 3 +

nums =

[2,7,11,15]

target =

9

```cpp
class Solution {
public:
    ListNode* addTwoNumbers(ListNode* l1, ListNode* l2) {
        ListNode* dummyHead = new ListNode(0);
        ListNode* tail = dummyHead;
        int carry = 0;

        while (l1 != nullptr || l2 != nullptr || carry != 0) {
            int digit1 = (l1 != nullptr) ? l1->val : 0;
            int digit2 = (l2 != nullptr) ? l2->val : 0;

            int sum = digit1 + digit2 + carry;
            int digit = sum % 10;
            carry = sum / 10;

            ListNode* newNode = new ListNode(digit);
            tail->next = newNode;
            tail = tail->next;

            l1 = (l1 != nullptr) ? l1->next : nullptr;
            l2 = (l2 != nullptr) ? l2->next : nullptr;
        }

        ListNode* result = dummyHead->next;
        delete dummyHead;
        return result;
    }
};
```

```cpp
class Solution {
public:
    ListNode* addTwoNumbers(ListNode* l1, ListNode* l2) {
        ListNode* dummyHead = new ListNode(0);
        ListNode* tail = dummyHead;
        int carry = 0;

        while (l1 != nullptr || l2 != nullptr || carry != 0) {
            int digit1 = (l1 != nullptr) ? l1->val : 0;
            int digit2 = (l2 != nullptr) ? l2->val : 0;

            int sum = digit1 + digit2 + carry;
            int digit = sum % 10;
            carry = sum / 10;

            ListNode* newNode = new ListNode(digit);
            tail->next = newNode;
            tail = tail->next;

            l1 = (l1 != nullptr) ? l1->next : nullptr;
```

Saved

Testcase | Test Result

Case 1 Case 2 Case 3 +

l1 =

[2,4,3]

l2 =

[5,6,4]

```cpp
class Solution {
public:
    int lengthOfLongestSubstring(string s) {
        int n = s.length();
        int maxLength = 0;
        unordered_set<char> charSet;
        int left = 0;

        for (int right = 0; right < n; right++) {
            if (charSet.count(s[right]) == 0) {
                charSet.insert(s[right]);
                maxLength = max(maxLength, right - left + 1);
            } else {
                while (charSet.count(s[right])) {
                    charSet.erase(s[left]);
                    left++;
                }
                charSet.insert(s[right]);
            }
        }

        return maxLength;
    }
};
```

Saved

Ln 1, Col 1

Testcase | Test Result

Case 1 Case 2 Case 3 +

s =

"abcabcbb"

```cpp
// Brute Force:
                // 1.Merge Both Array
               // 2.Sort them
              // 3.Find Median
             // TIME COMPLEXITY: O(n)+O(nlogn)+O(n)
             // SPACE COMPLEXITY: O(1)

class Solution {
public:
    double findMedianSortedArrays(vector<int>& nums1, vector<int>& nums2) {
       // Initialization some neccessary variables
        vector<int>v;

        // store the array in the new array
        for(auto num:nums1)   // O(n1)
            v.push_back(num);

        for(auto num:nums2)  // O(n2)
            v.push_back(num);

        // Sort the array to find the median
        sort(v.begin(),v.end());  // O(nlogn)

        // Find the median and Return it
        int n=v.size();  // O(n)

        return n%2?v[n/2]:(v[n/2-1]+v[n/2])/2.0;
    }
};
```

```cpp
// Brute Force:
                // 1.Merge Both Array
               // 2.Sort them
              // 3.Find Median
             // TIME COMPLEXITY: O(n)+O(nlogn)+O(n)
             // SPACE COMPLEXITY: O(1)

class Solution {
public:
    double findMedianSortedArrays(vector<int>& nums1, vector<int>& nums2) {
       // Initialization some neccessary variables
        vector<int>v;

        // store the array in the new array
        for(auto num:nums1)   // O(n1)
            v.push_back(num);

        for(auto num:nums2)  // O(n2)
```

Saved

Testcase | >_ Test Result

Case 1 Case 2 +

nums1 =

[1,3]

nums2 =

[2]

```cpp
class Solution {
private:
    bool check(string &s, int i, int j){
        while(i<j){
            if(s[i] != s[j]){
                return false;
            }
            i++;
            j--;
        }
        return true;
    }
public:
    string longestPalindrome(string s) {
        int n = s.size();
        int starting_index = 0;
        int max_len = 0;
        for(int i=0; i<n; i++){
            for(int j=i; j<n; j++){
                if(check(s, i, j)){
                    if(j-i+1 > max_len){
                        max_len = j-i+1;
                        starting_index = i;
                    }
                }
            }
        }
        return s.substr(starting_index, max_len);
    }
};
```

```cpp
class Solution {
private:
    bool check(string &s, int i, int j){
        while(i<j){
            if(s[i] != s[j]){
                return false;
            }
            i++;
            j--;
        }
        return true;
    }
public:
    string longestPalindrome(string s) {
        int n = s.size();
        int starting_index = 0;
        int max_len = 0;
        for(int i=0; i<n; i++){
```

Saved

Testcase | Test Result

Case 1 Case 2 +

s =

"babad"

C++ Auto

```cpp

class Solution {
public:

    string convert(string s, int numRows) {

    if(numRows <= 1) return s;

    vector<string>v(numRows, "");

    int j = 0, dir = -1;

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

        if(j == numRows - 1 || j == 0) dir *= (-1);

        v[j] += s[i];

        if(dir == 1) j++;

        else j--;
    }

    string res;

    for(auto &it : v) res += it;

    return res;

    }
};
```

```cpp

class Solution {
public:

    string convert(string s, int numRows) {

    if(numRows <= 1) return s;

    vector<string>v(numRows, "");

    int j = 0, dir = -1;

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

        if(j == numRows - 1 || j == 0) dir *= (-1);

```

Saved

Testcase | Test Result

Case 1 Case 2 Case 3 +

s =

"PAYPALISHIRING"

numRows =

3

```cpp
class Solution {
public:
    bool isMatch(string s, string p) {
        int n = s.length(), m = p.length();
        bool dp[n+1][m+1];
        memset(dp, false, sizeof(dp));
        dp[0][0] = true;

        for(int i=0; i<=n; i++){
            for(int j=1; j<=m; j++){
                if(p[j-1] == '*'){
                    dp[i][j] = dp[i][j-2] || (i > 0 && (s[i-1] == p[j-2] || p[j-2] == '.') && dp[i-1][j]);
                }
                else{
                    dp[i][j] = i > 0 && dp[i-1][j-1] && (s[i-1] == p[j-1] || p[j-1] == '.');
                }
            }
        }
```

Saved

Testcase | Test Result

Case 1 Case 2 Case 3 +

s =

"aa"

p =

"a"