LEETCODE STRING QUESTIONS:-

PROBLEM 1 : REMOVE OUTER PARANTHESIS:- [1021]

```
Input: s = "(()())(())"
Output: "()()()"
```

CODE-

```
class Solution {
public:
    string removeOuterParentheses(string s) {
        int count = 0;
        string res;
        for(char ch : s) {
            if(ch == '(') {
                if(count > 0) res += ch;
                count++;
            } else {
                count--;
                if(count > 0) res += ch;
            }
        }
        return res;
    }
};
```

PROBLEM 2 : REVERSE WORDS IN A STRING [151]

```
Input: s = "the sky is blue"
Output: "blue is sky the"
```

CODE-

```
class Solution {
public:
    string reverseWords(string s) {
        reverse(s.begin(), s.end());  // Step 1: Reverse whole string
        int i = 0, n = s.size();
        string result;

        while (i < n) {
            // Skip spaces
            while (i < n && s[i] == ' ') i++;

            // Find a word
            int start = i;
            while (i < n && s[i] != ' ') i++;
            int end = i;

            if (start < end) {
                string word = s.substr(start, end - start);
                reverse(word.begin(), word.end());
                if (!result.empty()) result += " ";
                result += word;
            }
        }

        return result;
```

```
    }
};
```

PROBLEM 3 : LARGEST ODD NUMBER IN A STRING [1903]-

Example 1:

```
Input: num = "52"
Output: "5"
Explanation: The only non-empty substrings are "5", "2", and "52". "5" is the o
nly odd number.
```

Example 2:

```
Input: num = "4206"
Output: ""
Explanation: There are no odd numbers in "4206".
```

Example 3:

```
Input: num = "35427"
Output: "35427"
Explanation: "35427" is already an odd number.
```

CODE-

```
class Solution {
public:
    string largestOddNumber(string num) {
        for (int i = num.size() - 1; i >= 0; i--) {
            if ((num[i] - '0') % 2 == 1) { // -'0' ascii to int conversion ke liy kia
                return num.substr(0, i + 1);
            }
        }
```

```
        return "";
    }
};
```

PROBLEM 4 : LONGEST COMMON PREFIX [14]

```
Input: strs = ["flower","flow","flight"]
Output: "fl"
```

CODE-

```
    class Solution {
public:
    string longestCommonPrefix(vector<string>& strs) {
        if (strs.empty()) return "";

        string ans = strs[0]; //string ki first string ko base liya jisse sbko compare
krenge

        for (int i = 1; i < strs.size(); i++) {
            int j = 0;
            while (j < ans.size() && j < strs[i].size() && ans[j] == strs[i][j]) {
                j++;
            }
            ans = ans.substr(0, j);
            if (ans == "") return "";
        }

        return ans;
    }
};
```

PROBLEM 5 : ISOMORPHIC STRINGS [205]

Example 1:

Input: s = "egg", t = "add"

Output: true

Explanation:

The strings `s` and `t` can be made identical by:

- Mapping `'e'` to `'a'`
- Mapping `'g'` to `'d'`

Example 2:

Input: s = "foo", t = "bar"

Output: false

Explanation:

The strings `s` and `t` cannot be made identical because `'o'` would need to be mapped to both `'a'` and `'r'`.

Example 3:

Input: s = "paper", t = "title"

Output: true

CODE-

```
class Solution {
public:
    bool isIsomorphic(string s, string t) {
        if (s.size() != t.size()) return false;

        unordered_map<char, char> m1, m2;
```

```
        for (int i = 0; i < s.size(); i++) {
            if (m1.count(s[i]) && m1[s[i]] != t[i]) return false;
            if (m2.count(t[i]) && m2[t[i]] != s[i]) return false;

            m1[s[i]] = t[i];
            m2[t[i]] = s[i];
        }

        return true;
    }
};
```

PROBLEM 6 : **check whether one string is a rotation of another [796]**

If we can achieve the goal then return true otherwise false.

Example 1:

```
Input: s = "abcde", goal = "cdeab"
Output: true
```

Example 2:

```
Input: s = "abcde", goal = "abced"
Output: false
```

CODE-

✅ **Key idea (super simple):**

> s ko 2 baar jod lo → s + s
>
> check karo **goal usmein hai ya nahi**!

```
class Solution {
public:
    bool rotateString(string s, string goal) {
        if (s.length() != goal.length()) {
            return false;
        }
        return (s + s).find(goal) != string::npos;
    }
};
```

PROBLEM 7 : ANAGRAM OR NOT [242]

Example 1:

Input: s = "anagram", t = "nagaram"

Output: true

Example 2:

Input: s = "rat", t = "car"

Output: false

CODE-

```
class Solution {
public:
    bool isAnagram(string s, string t) {
    if(s.size() != t.size()) return false;
    vector<int> count(26, 0);
    for(char ch : s) count[ch - 'a']++;
    for(char ch : t) count[ch - 'a']--;
    for(int c : count) if(c != 0) return false;
    return true;
```

```
}
};
```

PROBLEM 8- IMPLEMENT ATOI [8]

string to int

Example 1:

Input: s = "42"

Output: 42

Explanation:

```
The underlined characters are what is read in and the caret is the current read
er position.
Step 1: "42" (no characters read because there is no leading whitespace)
         ^
Step 2: "42" (no characters read because there is neither a '-' nor '+')
         ^
Step 3: "42" ("42" is read in)
           ^
```

Example 2:

Input: s = " -042"

Output: -42

Explanation:

```
Step 1: " -042" (leading whitespace is read and ignored)
          ^
Step 2: " -042" ('-' is read, so the result should be negative)
           ^
Step 3: " -042" ("042" is read in, leading zeros ignored in the result)
             ^
```

Example 3:

Input: s = "1337c0d3"

Output: 1337

Explanation:

```
Step 1: "1337c0d3" (no characters read because there is no leading whitespac
e)
        ^
Step 2: "1337c0d3" (no characters read because there is neither a '-' nor '+')
        ^
Step 3: "1337c0d3" ("1337" is read in; reading stops because the next charact
er is a non-digit)
            ^
```

Example 4:

Input: s = "0-1"

Output: 0

Explanation:

```
Step 1: "0-1" (no characters read because there is no leading whitespace)
        ^
Step 2: "0-1" (no characters read because there is neither a '-' nor '+')
        ^
Step 3: "0-1" ("0" is read in; reading stops because the next character is a no
n-digit)
         ^
```

Example 5:

Input: s = "words and 987"

Output: 0

Explanation:

Reading stops at the first non-digit character 'w'.

CODE -

```
int myAtoi(string s) {
    int i = 0, n = s.size();
    long num = 0;
    int sign = 1;

    // 1. Skip spaces
    while (i < n && s[i] == ' ') i++;

    // 2. Sign
    if (i < n && (s[i] == '+' || s[i] == '-')) {
        sign = (s[i] == '-') ? -1 : 1;
        i++;
    }

    // 3. Digits
    while (i < n && isdigit(s[i])) {
        num = num * 10 + (s[i] - '0');

        // 4. Clamp
        if (sign == 1 && num >= INT_MAX) return INT_MAX;
        if (sign == -1 && -num <= INT_MIN) return INT_MIN;

        i++;
    }

    return (int)(sign * num);
}
```

✅ Logic

⭐ 4 step mein kaam ho jayega:

1️⃣ Spaces hatao

2️⃣ **Sign dekho** (+ / -)

3️⃣ **Digit padho** jab tak milte rahe

4️⃣ **Overflow check karo** (INT_MIN / INT_MAX)

PROBLEM 9- **Sum of Beauty of all substring[1781]**

✅ **Ye kya problem hai?**

> Tumhe string di hai. Tum uske saare substrings banao. Har substring ka beauty nikaalo. Phir sab beauty ka sum return karo.

✅ **Beauty of a substring = frequency difference**

> For that substring:
>
> **Beauty = max frequency - min frequency (excluding zero)**

Example 1:

```
Input: s = "aabcb"
Output: 5
Explanation:The substrings with non-zero beauty are ["aab","aabc","aabcb","abcb","bcb"], each with beauty equal to 1.
```

Example 2:

```
Input: s = "aabcbaa"
Output: 17
```

✅ Example 1

```
s = "aabcb"
```

Saare substrings dekho:

- "a" → freq {a:1} → beauty = 1 - 1 = 0
- "aa" → {a:2} → beauty = 2 - 2 = 0
- "aab" → {a:2, b:1} → beauty = 2 - 1 = 1
- "aabc" → {a:2, b:1, c:1} → 2 - 1 = 1
- "aabcb" → {a:2, b:2, c:1} → 2 - 1 = 1
- etc.

CODE:

```
int beautySum(string s) {
    int n = s.size();
    int ans = 0;

    for(int i = 0; i < n; i++) {
        int freq[26] = {0};

        for(int j = i; j < n; j++) {
            freq[s[j] - 'a']++;

            int maxFreq = 0;
            int minFreq = INT_MAX;

            for(int k = 0; k < 26; k++) {
                if(freq[k] > 0) {
                    maxFreq = max(maxFreq, freq[k]);
                    minFreq = min(minFreq, freq[k]);
                }
            }
```

```
            ans += (maxFreq - minFreq);
        }
    }

    return ans;
}
```

✅ Step by Step

⭐ For each starting index i:

- freq array zero karo.

⭐ For each ending index j ≥ i:

- s[j] ko freq mein add karo.
- freq array mein max aur min (>0) dhundo.
- answer += (max - min)