**Question 1**

Given two strings s and t, determine if they are isomorphic.

Two strings s and t are isomorphic if the characters in s can be replaced to get t.

All occurrences of a character must be replaced with another character while preserving the order of characters. No two characters may map to the same character, but a character may map to itself.

**Example 1:**

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

**Output:** true

Sol: #include <iostream>

#include <unordered\_map>

using namespace std;

bool isIsomorphic(string s, string t) {

    if (s.length() != t.length())

        return false;

    unordered\_map<char, char> s\_to\_t;

    unordered\_map<char, char> t\_to\_s;

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

        char ch\_s = s[i];

        char ch\_t = t[i];

        if (s\_to\_t.count(ch\_s) && s\_to\_t[ch\_s] != ch\_t)

            return false;

        if (t\_to\_s.count(ch\_t) && t\_to\_s[ch\_t] != ch\_s)

            return false;

        s\_to\_t[ch\_s] = ch\_t;

        t\_to\_s[ch\_t] = ch\_s;

    }

    return true;

}

int main() {

    string s = "egg";

    string t = "add";

    if (isIsomorphic(s, t))

        cout << "True" << endl;

    else

        cout << "False" << endl;

    return 0;

}

Output:True

**Question 2**

Given a string num which represents an integer, return true if num is a ***strobogrammatic number***.

A **strobogrammatic number** is a number that looks the same when rotated 180 degrees (looked at upside down).

**Example 1:**

**Input:** num = "69"

**Output:**

true

sol: #include <iostream>

using namespace std;

bool isStrobogrammatic(string num) {

    int left = 0;

    int right = num.length() - 1;

    while (left <= right) {

        char ch\_left = num[left];

        char ch\_right = num[right];

        if ((ch\_left == '6' && ch\_right == '9') ||

            (ch\_left == '9' && ch\_right == '6') ||

            (ch\_left == '0' && ch\_right == '0') ||

            (ch\_left == '1' && ch\_right == '1') ||

            (ch\_left == '8' && ch\_right == '8')) {

            left++;

            right--;

        } else {

            return false;

        }

    }

    return true;

}

int main() {

    string num = "69";

    if (isStrobogrammatic(num))

        cout << "true" << endl;

    else

        cout << "false" << endl;

    return 0;

}

Output:True

**Question 3**

Given two non-negative integers, num1 and num2 represented as string, return the sum of num1 and num2 as a string.

You must solve the problem without using any built-in library for handling large integers (such as BigInteger). You must also not convert the inputs to integers directly.

**Example 1:**

**Input:** num1 = "11", num2 = "123"

**Output:**

"134"

Sol: #include <iostream>

#include <algorithm>

using namespace std;

string addStrings(string num1, string num2) {

    int ptr1 = num1.length() - 1;

    int ptr2 = num2.length() - 1;

    int carry = 0;

    string result = "";

    while (ptr1 >= 0 || ptr2 >= 0) {

        int d1 = ptr1 >= 0 ? num1[ptr1] - '0' : 0;

        int d2 = ptr2 >= 0 ? num2[ptr2] - '0' : 0;

        int digit\_sum = d1 + d2 + carry;

        int current\_digit = digit\_sum % 10;

        carry = digit\_sum / 10;

        result += to\_string(current\_digit);

        ptr1--;

        ptr2--;

    }

    if (carry > 0)

        result += to\_string(carry);

    reverse(result.begin(), result.end());

    return result;

}

int main() {

    string num1 = "11";

    string num2 = "123";

    string sum = addStrings(num1, num2);

    cout << sum << endl;

    return 0;

}

Output: 134

**Question 4**

Given a string s, reverse the order of characters in each word within a sentence while still preserving whitespace and initial word order.

**Example 1:**

**Input:** s = "Let's take LeetCode contest"

**Output:** "s'teL ekat edoCteeL tsetnoc"

Sol: #include <iostream>

#include <sstream>

#include <vector>

#include <algorithm>

using namespace std;

string reverseWords(string s) {

    stringstream ss(s);

    vector<string> words;

    string word;

    // Split the string into words

    while (ss >> word) {

        words.push\_back(word);

    }

    // Reverse the characters in each word

    for (string& w : words) {

        reverse(w.begin(), w.end());

    }

    // Join the reversed words back together

    string result = "";

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

        result += words[i];

        if (i < words.size() - 1) {

            result += " ";

        }

    }

    return result;

}

int main() {

    string s = "Let's take LeetCode contest";

    string reversed = reverseWords(s);

    cout << reversed << endl;

    return 0;

}

Output: s'teL ekat edoCteeL tsetnoc

**Question 5**

Given a string s and an integer k, reverse the first k characters for every 2k characters counting from the start of the string.

If there are fewer than k characters left, reverse all of them. If there are less than 2k but greater than or equal to k characters, then reverse the first k characters and leave the other as original.

**Example 1:**

**Input:** s = "abcdefg", k = 2

**Output:**

"bacdfeg"

Sol: #include <iostream>

#include <algorithm>

using namespace std;

string reverseStr(string s, int k) {

    char\* arr = &s[0]; // Convert string to character array

    int n = s.length();

    for (int i = 0; i < n; i += 2 \* k) {

        int left = i;

        int right = min(i + k - 1, n - 1);

        while (left < right) {

            swap(arr[left], arr[right]);

            left++;

            right--;

        }

    }

    return string(arr); // Convert character array back to string

}

int main() {

    string s = "abcdefg";

    int k = 2;

    string reversed = reverseStr(s, k);

    cout << reversed << endl;

    return 0;

}

Output: bacdfeg

**Question 6**

Given two strings s and goal, return true *if and only if* s *can become* goal *after some number of* ***shifts*** *on* s.

A **shift** on s consists of moving the leftmost character of s to the rightmost position.

* For example, if s = "abcde", then it will be "bcdea" after one shift.

**Example 1:**

**Input:** s = "abcde", goal = "cdeab"

**Output:**

true

sol: #include <iostream>

#include <algorithm>

using namespace std;

string reverseStr(string s, int k) {

    char\* arr = &s[0]; // Convert string to character array

    int n = s.length();

    for (int i = 0; i < n; i += 2 \* k) {

        int left = i;

        int right = min(i + k - 1, n - 1);

        while (left < right) {

            swap(arr[left], arr[right]);

            left++;

            right--;

        }

    }

    return string(arr); // Convert character array back to string

}

int main() {

    string s = "abcdefg";

    int k = 2;

    string reversed = reverseStr(s, k);

    cout << reversed << endl;

    return 0;

}

Output:True

**Question 7**

Given two strings s and t, return true if they are equal when both are typed into empty text editors. '#' means a backspace character.

Note that after backspacing an empty text, the text will continue empty.

**Example 1:**

**Input:** s = "ab#c", t = "ad#c"

**Output:** true

**Explanation:**

Both s and t become "ac".

Sol: #include <iostream>

#include <stack>

using namespace std;

bool backspaceCompare(string s, string t) {

    stack<char> stackS, stackT;

    for (char c : s) {

        if (c != '#')

            stackS.push(c);

        else if (!stackS.empty())

            stackS.pop();

    }

    for (char c : t) {

        if (c != '#')

            stackT.push(c);

        else if (!stackT.empty())

            stackT.pop();

    }

    while (!stackS.empty() && !stackT.empty()) {

        if (stackS.top() != stackT.top())

            return false;

        stackS.pop();

        stackT.pop();

    }

    return stackS.empty() && stackT.empty();

}

int main() {

    string s = "ab#c";

    string t = "ad#c";

    if (backspaceCompare(s, t))

        cout << "true" << endl;

    else

        cout << "false" << endl;

    return 0;

}

Output:True

**Question 8**

You are given an array coordinates, coordinates[i] = [x, y], where [x, y] represents the coordinate of a point. Check if these points make a straight line in the XY plane.

**Example 1:**

**Input:** coordinates = [[1,2],[2,3],[3,4],[4,5],[5,6],[6,7]]

**Output:** true

Sol: #include <iostream>

#include <vector>

using namespace std;

bool checkStraightLine(vector<vector<int>>& coordinates) {

    int n = coordinates.size();

    if (n < 2)

        return true;

    int x1 = coordinates[0][0], y1 = coordinates[0][1];

    int x2 = coordinates[1][0], y2 = coordinates[1][1];

    float slope = static\_cast<float>(y2 - y1) / (x2 - x1);

    for (int i = 2; i < n; i++) {

        x1 = coordinates[i - 1][0];

        y1 = coordinates[i - 1][1];

        x2 = coordinates[i][0];

        y2 = coordinates[i][1];

        float currentSlope = static\_cast<float>(y2 - y1) / (x2 - x1);

        if (currentSlope != slope)

            return false;

    }

    return true;

}

int main() {

    vector<vector<int>> coordinates = {{1, 2}, {2, 3}, {3, 4}, {4, 5}, {5, 6}, {6, 7}};

    if (checkStraightLine(coordinates))

        cout << "true" << endl;

    else

        cout << "false" << endl;

    return 0;

}

Output:True