**Problem 1: Given an array of strings, group anagrams together**

**Code:**

#include <bits/stdc++.h>

using namespace std;

vector<vector<string>> groupana(vector<string>& str) {

    unordered\_map<string, vector<string>> mp;

    for (string x : str) {

        string k = x;

        sort(k.begin(), k.end());

        mp[k].push\_back(x);

    }

    vector<vector<string>> res;

    for (auto i : mp) {

        res.push\_back(i.second);

    }

    return res;

}

int main(){

    vector<string> str = {"tame", "meat", "snow", "wons", "care", "star", "race", "rats"};

    vector<vector<string>> ans = groupana(str);

    for(auto x : ans){

        for(auto y : x){

            cout << y << " ";

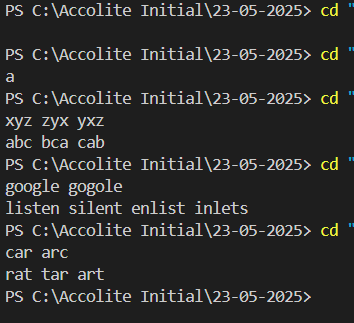
        }

        cout << endl;

    }

}

**Output:**

****

**Problem 3: Given a string containing just the bracket characters, determine if the input string is valid**

**Code:**

#include <bits/stdc++.h>

using namespace std;

bool validpar(string s){

    unordered\_map<char, char> mp = {{')','('}, {'}', '{'}, {']', '['}};

    stack<char> st;

    for(char x : s){

        if(x=='('||x=='{'||x=='['){

            st.push(x);

        }

        else{

            if(st.empty() || st.top()!=mp[x]){

                return false;

            }

            st.pop();

        }

    }

    return st.empty();

}

int main(){

    string s;

    cout << "Enter string: ";

    cin >> s;

    if(validpar(s)){

        cout << "Valid";

    }

    else{

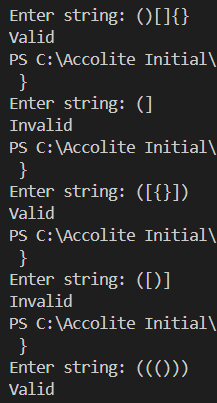
        cout << "Invalid";

    }

    return 0;

}

**Output:**

****

**Problem 4: Given two strings ss and tt, return the smallest substring in ss that contains all the characters in tt. If no such substring exists, return an empty string “”.**

**Code:**

#include <bits/stdc++.h>

using namespace std;

string smallestsun(string s, string t){

    unordered\_map<char, int> tar;

    unordered\_map<char, int> curr;

    for(char x : t){

        tar[x]++;

    }

    int l = 0, r = 0, minl = 1000, start = 0;

    int val = 0;

    while(r<s.length()){

        char x = s[r++];

        if(tar.find(x)!=tar.end()){

            curr[x]++;

            if(curr[x]==tar[x]){

                val++;

            }

        }

        while(val==tar.size()){

            if(r-l<minl){

                minl = r - l;

                start = l;

            }

            char y = s[l++];

            if(tar.find(y)!=tar.end()){

                if(curr[y]==tar[y]){

                    val--;

                }

                curr[y]--;

            }

        }

    }

    if(minl==1000){

        return "";

    }

    string res = s.substr(start, minl);

    return res;

}

int main(){

    string s, t;

    cout << "Enter string: ";

    cin >> s;

    cout << "Enter target: ";

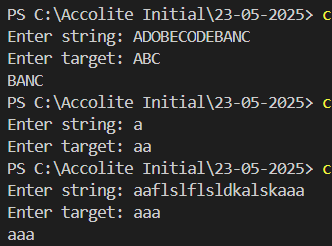
    cin >> t;

    cout << smallestsun(s, t);

    return 0;

}

**Output:**

****

**Problem 5:**

**Two strings are considered close if you can swap letters or change the frequency of any letter to match the other string. Determine if two given strings are close.**

#include <bits/stdc++.h>

using namespace std;

bool closestring(string s, string t){

    unordered\_set<char> s1(s.begin(), s.end());

    unordered\_set<char> s2(t.begin(), t.end());

    return (s1==s2 && s.length() == t.length());

}

int main(){

    string s, t;

    cout << "Enter string: ";

    cin >> s;

    cout << "Enter target: ";

    cin >> t;

    if(closestring(s,t)){

        cout << "True";

    }

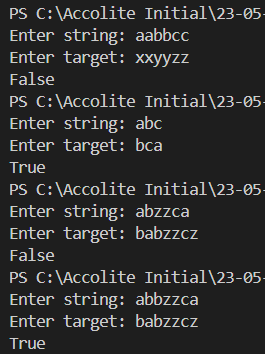
    else{

        cout << "False";

    }

}

**Output:**

****

**Problem 6: Given two strings ss and tt, determine if tt is an anagram of a substring of ss. In other words, check if there exists a substring in ss that is an anagram of tt.**

**Code:**

#include <bits/stdc++.h>

using namespace std;

bool containsAnagram(string s, string t) {

    if (t.length() > s.length()) return false;

    vector<int> tar(26, 0), curr(26, 0);

    for (char c : t){

        tar[c - 'a']++;

    }

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

        curr[s[i] - 'a']++;

        if (i >= t.length()) {

            curr[s[i - t.length()] - 'a']--;

        }

        if (curr == tar) return true;

    }

    return false;

}

int main() {

    string s, t;

    cout << "Enter string: ";

    cin >> s;

    cout << "Enter pattern: ";

    cin >> t;

    if (containsAnagram(s, t)) {

        cout << "True";

    } else {

        cout << "False";

    }

    return 0;

}

**Output:**

****