Question 2 -

Given a string str, return the longest palindromic substring in str.

Code -

#include <iostream>

#include <string>

using namespace std;

string lstPalin(string str) {

int n = str.size(), start = 0, maxlen = 1;

for (int i = 0; i < n;) {

int left = i, right = i;

while (right + 1 < n && str[right + 1] == str[i]) right++;

i = right + 1;

while (left - 1 >= 0 && right + 1 < n && str[left - 1] == str[right + 1]) {

left--; right++;

}

if (right - left + 1 > maxlen) {

start = left;

maxlen = right - left + 1;

}

}

return str.substr(start, maxlen);

}

int main() {

string str;

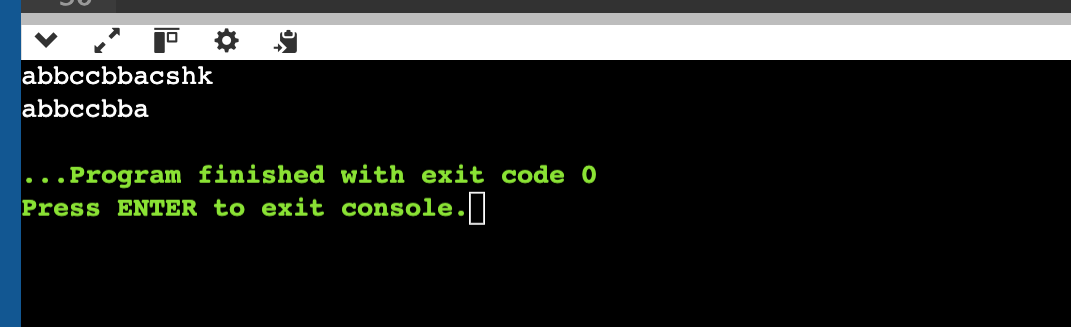
cin >> str;

cout<< lstPalin(str);

return 0;

}

Output-



Question 3 -

Valid Parentheses

Given a string containing just the characters "(",")","{","}","[",and "]", determine if the input string is valid.

Code -

#include <iostream>

#include <stack>

#include <string>

using namespace std;

bool isValid(string str) {

stack<char> st;

for (char ch : str) {

if (ch == '(' || ch == '{' || ch == '[') {

st.push(ch);

} else {

if (st.empty()) return false;

char top = st.top();

if ((ch == ')' && top != '(') ||

(ch == '}' && top != '{') ||

(ch == ']' && top != '[')) {

return false;

}

st.pop();

}

}

return st.empty();

}

int main() {

string str;

cin >> str;

if (isValid(str)) {

cout << "True" << endl;

} else {

cout << "False" << endl;

}

return 0;

}

Output-



Question 4 -

Given two strings ss and rt, 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 minWindow(string ss, string tt) {

if (ss.length() < tt.length()) return "";

unordered\_map<char, int> need, window;

for (char ch : tt) need[ch]++;

int left = 0, right = 0, valid = 0, start = 0, len = INT\_MAX;

while (right < ss.size()) {

char ch = ss[right++];

if (need.count(ch)) {

window[ch]++;

if (window[ch] == need[ch]) valid++;

}

while (valid == need.size()) {

if (right - left < len) {

start = left;

len = right - left;

}

char d = ss[left++];

if (need.count(d)) {

if (window[d] == need[d]) valid--;

window[d]--;

}

}

}

return len == INT\_MAX ? "" : ss.substr(start, len);

}

int main() {

string ss, tt;

cout << "Enter string: ";

cin >> ss;

cout << "Enter target: ";

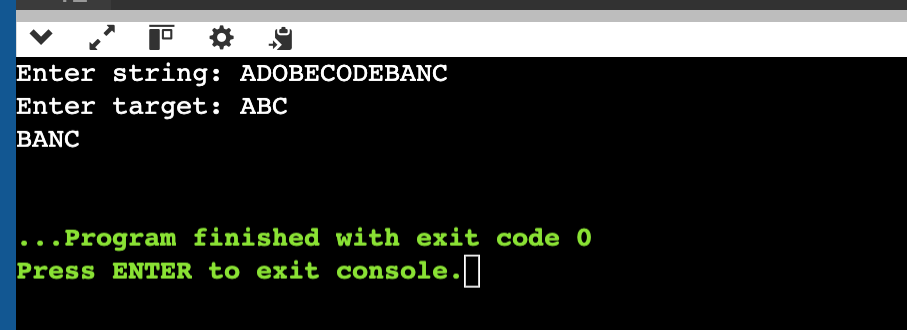
cin >> tt;

cout <<minWindow(ss, tt) << endl;

return 0;

}

Output-



Question 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.

Code -

#include <bits/stdc++.h>

using namespace std;

bool closestr(string str1, string str2) {

if (str1.size() != str2.size()) return false;

vector<int> freq1(26, 0), freq2(26, 0);

unordered\_set<char> s1, s2;

for (char c : str1) freq1[c - 'a']++, s1.insert(c);

for (char c : str2) freq2[c - 'a']++, s2.insert(c);

sort(freq1.begin(), freq1.end());

sort(freq2.begin(), freq2.end());

return freq1 == freq2 && s1 == s2;

}

int main() {

string s1, s2;

cin >> s1;

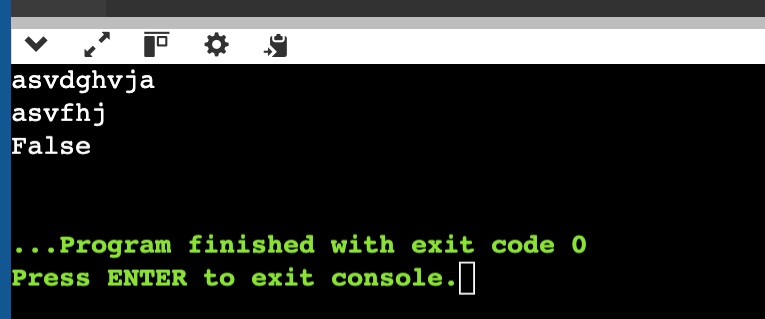
cin >> s2;

cout << (closestr(s1, s2) ? "True" : "False") << endl;

return 0;

}

Output-



Question 6 -

Given two strings ss and tt, determine if it 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 checkmatch(string t, string s) {

if (s.size() < t.size()) return false;

vector<int> target(26, 0), window(26, 0);

for (char c : t) target[c - 'a']++;

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

window[s[i] - 'a']++;

if (i >= t.size()) window[s[i - t.size()] - 'a']--;

if (window == target) return true;

}

return false;

}

int main() {

string s, t;

cin >> s;

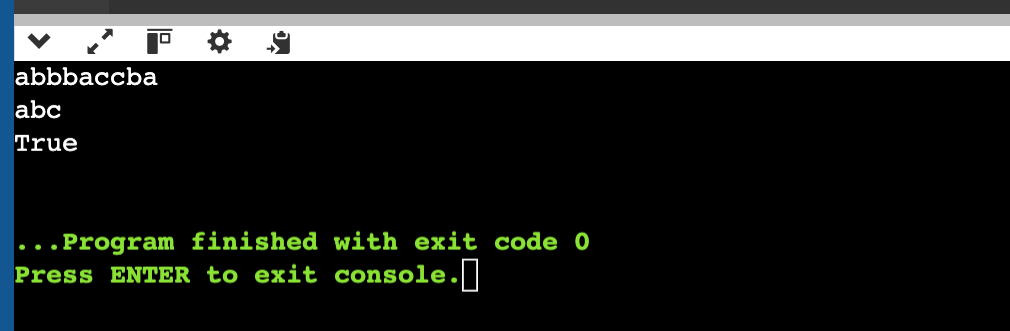
cin >> t;

cout << (checkmatch(t, s) ? "True" : "False") << endl;

return 0;

}

Output-



Question 7 -

Write a function to find the longest common prefix string amongst an array of strings.

If there is no common prefix, return an empty string.

Code -

#include <bits/stdc++.h>

using namespace std;

string lcp(vector<string>& strs) {

if (strs.empty()) return "";

string prefix = strs[0];

for (int i = 1; i < strs.size(); ++i) {

while (strs[i].find(prefix) != 0) {

prefix.pop\_back();

if (prefix.empty()) return "";

}

}

return prefix;

}

int main() {

int n;

cout << "Enter number of strings: ";

cin >> n;

vector<string> strs(n);

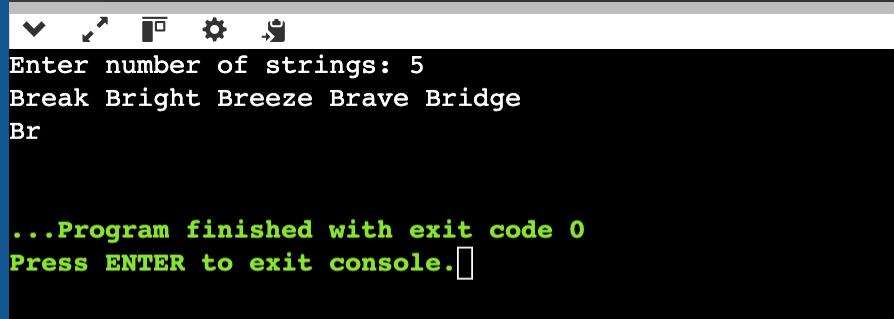
for (int i = 0; i < n; ++i) cin >> strs[i];

cout << lcp(strs) << endl;

return 0;

}

Output-



Question 8 -

Given two strings needle and haystack, return the index of the first occurrence of needle in haystack, or -1 if needle is not part of haystack.

Code -

#include <bits/stdc++.h>

using namespace std;

int strocc(string haystack, string needle) {

if (needle.empty()) return 0;

int n = haystack.size(), m = needle.size();

for (int i = 0; i <= n - m; ++i)

if (haystack.substr(i, m) == needle) return i;

return -1;

}

int main() {

string haystack, needle;

cin >> haystack;

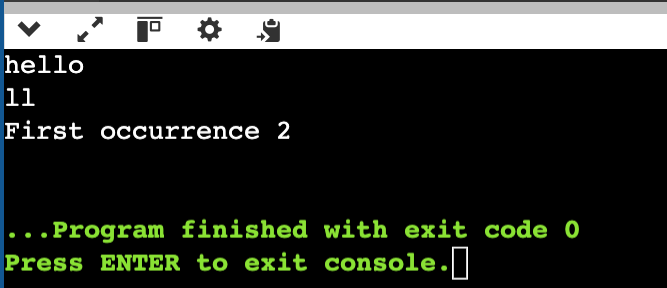
cin >> needle;

cout << "First occurrence " << strocc(haystack, needle) << endl;

return 0;

}

Output-



Question 9 -

Given a string s consisting of words and spaces, return the length of the last word in the string.

A word is a maximal substring consisting of non-space characters only.

Code -

#include <bits/stdc++.h>

using namespace std;

int lenlastword(string str) {

int n = str.length() - 1, len = 0;

while (n >= 0 && str[n] == ' ') n--;

while (n >= 0 && str[n] != ' ') len++, n--;

return len;

}

int main() {

string str;

cin.ignore();

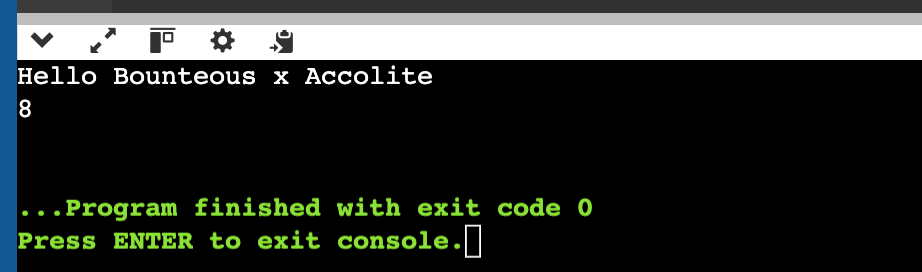
getline(cin, str);

cout << lenlastword(str) << endl;

return 0;

}

Output-



Question 10 -

Given a string paragraph and a string array of the banned words banned, return the most frequent word that is not banned. It is guaranteed there is at least one word that is not banned, and that the answer is unique.

The words in paragraph are case-insensitive and the answer should be returned in lowercase.

Code -

#include <bits/stdc++.h>

using namespace std;

string mostCommonWord(string paragraph, vector<string>& banned) {

unordered\_set<string> ban(banned.begin(), banned.end());

unordered\_map<string, int> count;

string word, res;

int maxfreq = 0;

for (char& c : paragraph) c = isalpha(c) ? tolower(c) : ' ';

istringstream iss(paragraph);

while (iss >> word) {

if (!ban.count(word)) {

count[word]++;

if (count[word] > maxfreq) {

maxfreq = count[word];

res = word;

}

}

}

return res;

}

int main() {

string paragraph;

int n;

cout << "Enter paragraph: ";

cin.ignore();

getline(cin, paragraph);

cout << "Banned words count: ";

cin >> n;

vector<string> banned(n);

cout << "Enter banned words:\n";

for (int i = 0; i < n; ++i) cin >> banned[i];

cout << "Most Common Non-Banned Word: " << mostCommonWord(paragraph, banned) << endl;

return 0;

}

Output-

