Question 1 (1)

#include<bits/stdc++.h>

using namespace std ;

int main() {

string input\_word ;

cout << "Enter Your String :- " ;

cin >> input\_word ;

vector<string> valid\_words = {"apple" , "application" , "grape" , "pineapple" , "banana"} ;

int max\_correct = 0 ;

int correct\_idx = -1 ;

int i = 0 ;

while(i < valid\_words.size()) {

if(input\_word.size() != valid\_words[i].size()) {

i++;

continue;

}

int current\_correct = 0 ;

for(int j = 0 ; j < input\_word.size() ; j++) {

if(input\_word[j] == valid\_words[i][j]) {

current\_correct++;

}

}

if(current\_correct > max\_correct ) {

max\_correct = current\_correct ;

correct\_idx = i ;

}

i++;

}

if(correct\_idx != -1) {

cout << "Correct word is " << valid\_words[correct\_idx] << endl ;

}

else {

cout << "No Closest Word Found" << endl;

}

}

Question – 1 (2)

#include<bits/stdc++.h>

using namespace std ;

int main() {

string order ;

string s ;

cout << "Enter the Order String :- " ;

cin >> order ;

cout << "Enter the Input String S :- " ;

cin >> s ;

unordered\_map<char , int> priority\_table ;

int priority = order.size();

for(char c : order) {

priority\_table[c] = priority ;

priority--;

}

int i = 0 ;

int j = 1 ;

while(i < j && j < s.size()) {

if(priority\_table[j] > priority\_table[i]) {

char temp = s[i] ;

s[i] = s[j] ;

s[j] = temp ;

i++ ;

j = i+1;

}

j++;

}

cout << "Output is :- " << s << endl;

}

Question-2(1)

#include<bits/stdc++.h>

using namespace std ;

string decimal\_to\_binary(int num) {

string ans = "" ;

while(num != 0) {

char curr = (num % 2 == 0 ? '0' : '1') ;

ans += curr ;

num /= 2 ;

}

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

return ans;

}

int main() {

int a;

int b ;

cout << "Enter Number - 1 :- " ;

cin >> a ;

cout << "Enter Number - 2 :- " ;

cin >> b ;

string n1 = decimal\_to\_binary(a) ;

string n2 = decimal\_to\_binary(b) ;

for(int i = 1 ; i <= 8 - n1.size(); i++) {

n1 = "0" + n1 ;

}

for(int i = 1 ; i <= 8 - n2.size(); i++) {

n2 = "0" + n2 ;

}

int flip = 0 ;

int no\_of\_bits = n1.size();

for(int i = 0 ; i < no\_of\_bits ; i++) {

if(n1[i] != n2[i]) {

flip++;

}

}

cout << "Flips Required are :- " << flip << endl;

}

Question 3(B)

#include <bits/stdc++.h>

using namespace std;

int findCutVertex (vector<vector<int>> &graph, int node) {

int min\_degree = INT\_MAX ;

int cut\_vertices = -1 ;

for(int i = 0 ; i < node ; i++) {

int curr\_degree = 0 ;

for(int j = 0 ; j < node ; j++) {

if(graph[i][j] != 9999 && graph[i][j] != 0) {

curr\_degree++;

}

}

if(curr\_degree < min\_degree) {

cut\_vertices = i ;

}

}

return cut\_vertices;

}

int main()

{

int node;

cout << "Enter The Number of Nodes :- ";

cin >> node;

vector<vector<int>> graph(node, vector<int>(node, 9999));

cout << "Enter The Adjacency Matrix :- " << endl;

for (int i = 0; i < node; i++)

{

for (int j = 0; j < node; j++)

{

cout << "Enter The Edge from " << i << " to " << j << " (9999 for no edge) :- ";

cin >> graph[i][j];

}

}

int cut\_vertices = findCutVertex(graph, node);

if(cut\_vertices != -1) {

cout << "The Cut Vertex Is " << cut\_vertices << endl ;

}

else {

cout << "All Vertex Having the Strong Connectiveity." << endl ;

}

return 0;

}

Question 4 :-

#include<bits/stdc++.h>

using namespace std ;

bool isPalindrome(string &s , int start , int end) {

string curr(s.begin() + start , s.end() + (end + 1)) ;

int size = end - start + 1;

for(int i = 0 ; i <= size / 2 ; i++) {

if(s[i] != s[size - i - 1]) {

return false ;

}

}

return true;

}

int Total\_Palindrome\_Subsequence(string &s) {

int total = 0 ;

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

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

bool ans = isPalindrome(s, i , j) ;

total += ans ;

}

}

return total ;

}

int main() {

string s ;

cout << "Enter the String :- " ;

cin >> s ;

int total\_Palindrome = Total\_Palindrome\_Subsequence(s) ;

cout <<"The Total Palindrome Subsequences are " << total\_Palindrome << endl;

return 0;

}