**Cryptography and 19115045**

**Network Security Lab Kunal Sachdeva**

**LAB-1 6th Sem CSE**

**1. Write a Program to find the frequency of occurrence of characters.**

**Code:**

#include <bits/stdc++.h>

using namespace std;

int main()

{

unordered\_map<char,int> freq;

string s;

cout<<"Enter a string: ";

getline(cin,s);

cout<<s<<endl;

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

{

if(s[i]!=' ')

freq[s[i]]++;

}

for(auto i:freq)

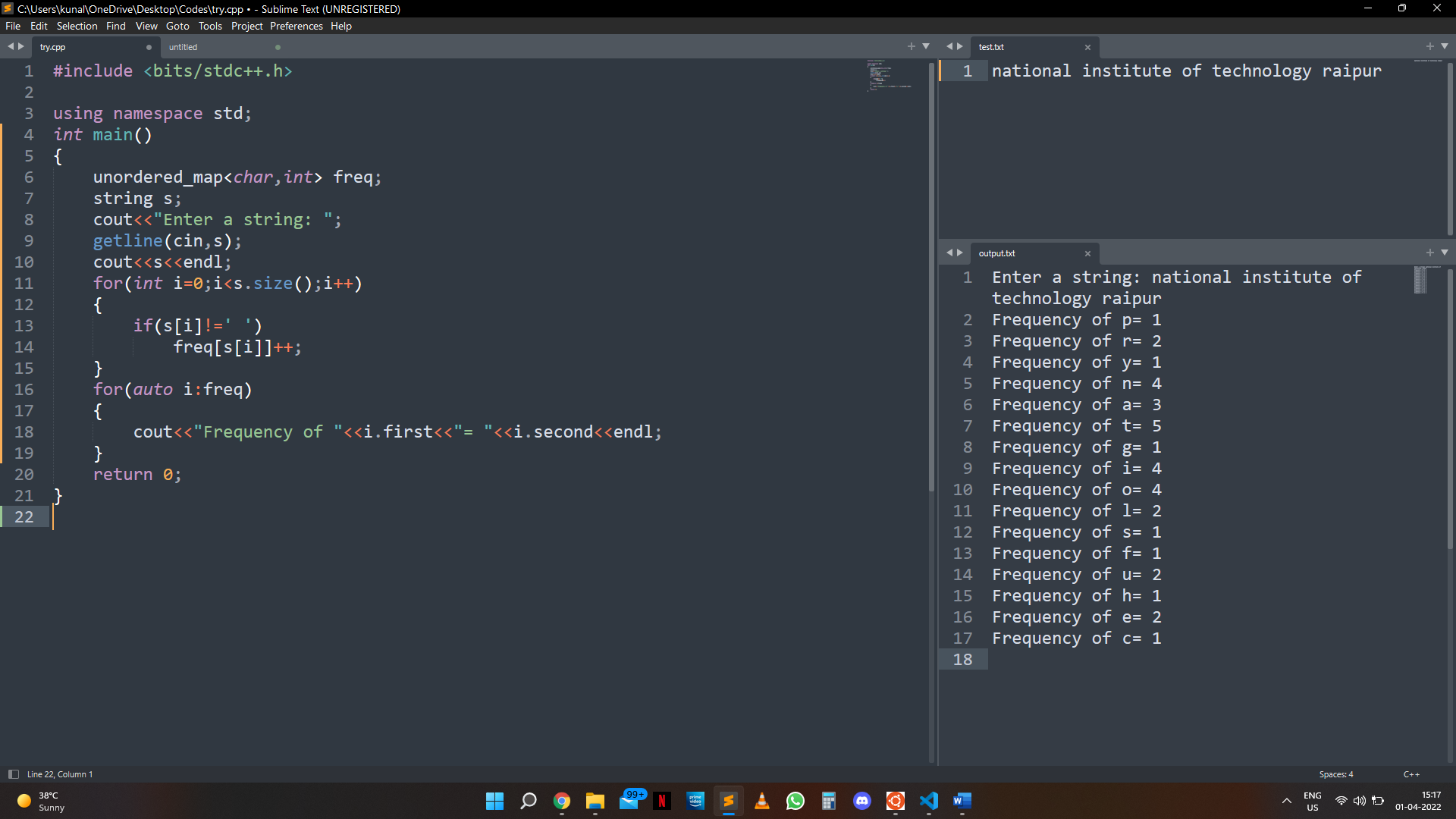
{

cout<<"Frequency of "<<i.first<<"= "<<i.second<<endl;

}

return 0;

}

**Output:**

**2. Write a program to implement the concept of Caesar Cipher.**

**Code:**

#include <bits/stdc++.h>

using namespace std;

string encrypt(string text, int shift)

{

string result="";

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

{

if(isupper(text[i]))

result += char(int(text[i]+shift-65)%26 +65);

else

result += char(int(text[i]+shift-97)%26 +97);

}

return result;

}

int main()

{

string text;

cout<<"Enter the Plain Text: ";

getline(cin,text);

cin>> text;

int shift = 4;

cout<<"Text: "<<text<<endl;

cout<<"Shift: "<<shift<<endl;

string encryptedText=encrypt(text,shift);

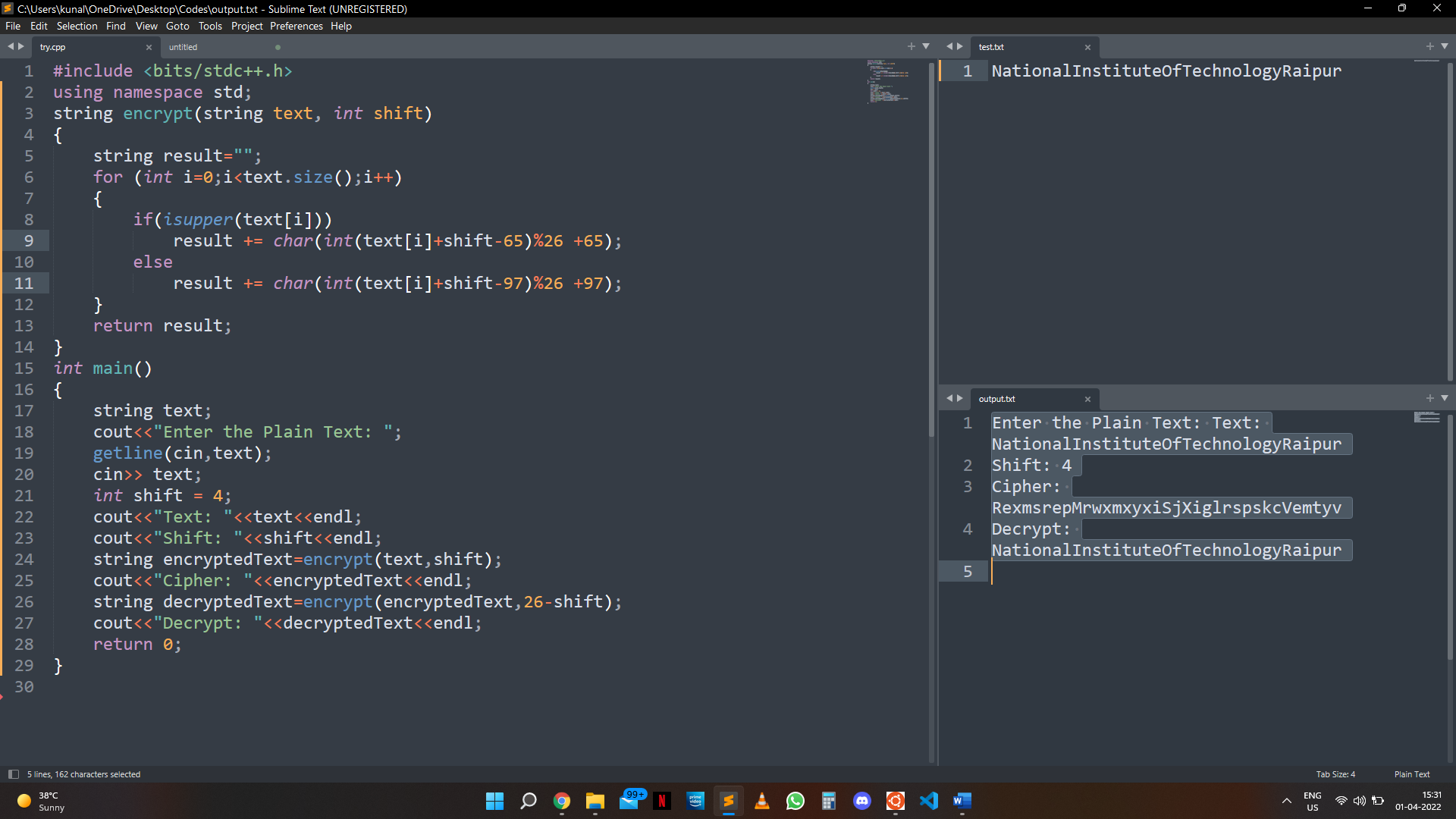
cout<<"Cipher: "<<encryptedText<<endl;

string decryptedText=encrypt(encryptedText,26-shift);

cout<<"Decrypt: "<<decryptedText<<endl;

return 0;

}

**Output:**

**3. Write a program to implement the concept of Playfair Cipher.**

**Code:**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define SIZE 30

void toLowerCase(char plain[], int ps)

{

int i;

for (i = 0; i < ps; i++) {

if (plain[i] > 64 && plain[i] < 91)

plain[i] += 32;

}

}

int removeSpaces(char\* plain, int ps)

{

int i, count = 0;

for (i = 0; i < ps; i++)

if (plain[i] != ' ')

plain[count++] = plain[i];

plain[count] = '\0';

return count;

}

void generateKeyTable(char key[], int ks, char keyT[5][5])

{

int i, j, k, flag = 0, \*dicty;

dicty = (int\*)calloc(26, sizeof(int));

for (i = 0; i < ks; i++) {

if (key[i] != 'j')

dicty[key[i] - 97] = 2;

}

dicty['j' - 97] = 1;

i = 0;

j = 0;

for (k = 0; k < ks; k++) {

if (dicty[key[k] - 97] == 2) {

dicty[key[k] - 97] -= 1;

keyT[i][j] = key[k];

j++;

if (j == 5) {

i++;

j = 0;

}

}

}

for (k = 0; k < 26; k++) {

if (dicty[k] == 0) {

keyT[i][j] = (char)(k + 97);

j++;

if (j == 5) {

i++;

j = 0;

}

}

}

}

void search(char keyT[5][5], char a, char b, int arr[])

{

int i, j;

if (a == 'j')

a = 'i';

else if (b == 'j')

b = 'i';

for (i = 0; i < 5; i++) {

for (j = 0; j < 5; j++) {

if (keyT[i][j] == a) {

arr[0] = i;

arr[1] = j;

}

else if (keyT[i][j] == b) {

arr[2] = i;

arr[3] = j;

}

}

}

}

int mod5(int a)

{

if (a < 0)

a += 5;

return (a % 5);

}

int prepare(char str[], int ptrs)

{

if (ptrs % 2 != 0) {

str[ptrs++] = 'z';

str[ptrs] = '\0';

}

return ptrs;

}

void encrypt(char str[], char keyT[5][5], int ps)

{

int i, a[4];

for (i = 0; i < ps; i += 2) {

search(keyT, str[i], str[i + 1], a);

if (a[0] == a[2]) {

str[i] = keyT[a[0]][mod5(a[1] + 1)];

str[i + 1] = keyT[a[0]][mod5(a[3] + 1)];

}

else if (a[1] == a[3]) {

str[i] = keyT[mod5(a[0] + 1)][a[1]];

str[i + 1] = keyT[mod5(a[2] + 1)][a[1]];

}

else {

str[i] = keyT[a[0]][a[3]];

str[i + 1] = keyT[a[2]][a[1]];

}

}

}

void encryptByPlayfairCipher(char str[], char key[])

{

char ps, ks, keyT[5][5];

ks = strlen(key);

ks = removeSpaces(key, ks);

toLowerCase(key, ks);

ps = strlen(str);

toLowerCase(str, ps);

ps = removeSpaces(str, ps);

ps = prepare(str, ps);

generateKeyTable(key, ks, keyT);

encrypt(str, keyT, ps);

}

void decrypt(char str[], char keyT[5][5], int ps)

{

int i, a[4];

for (i = 0; i < ps; i += 2) {

search(keyT, str[i], str[i + 1], a);

if (a[0] == a[2]) {

str[i] = keyT[a[0]][mod5(a[1] - 1)];

str[i + 1] = keyT[a[0]][mod5(a[3] - 1)];

}

else if (a[1] == a[3]) {

str[i] = keyT[mod5(a[0] - 1)][a[1]];

str[i + 1] = keyT[mod5(a[2] - 1)][a[1]];

}

else {

str[i] = keyT[a[0]][a[3]];

str[i + 1] = keyT[a[2]][a[1]];

}

}

}

void decryptByPlayfairCipher(char str[], char key[])

{

char ps, ks, keyT[5][5];

ks = strlen(key);

ks = removeSpaces(key, ks);

toLowerCase(key, ks);

ps = strlen(str);

toLowerCase(str, ps);

ps = removeSpaces(str, ps);

generateKeyTable(key, ks, keyT);

decrypt(str, keyT, ps);

}

int main()

{

char str[SIZE], key[SIZE];

strcpy(key, "Monarchy");

printf("Key text: %s\n", key);

strcpy(str, "instruments");

printf("Plain text: %s\n", str);

encryptByPlayfairCipher(str, key);

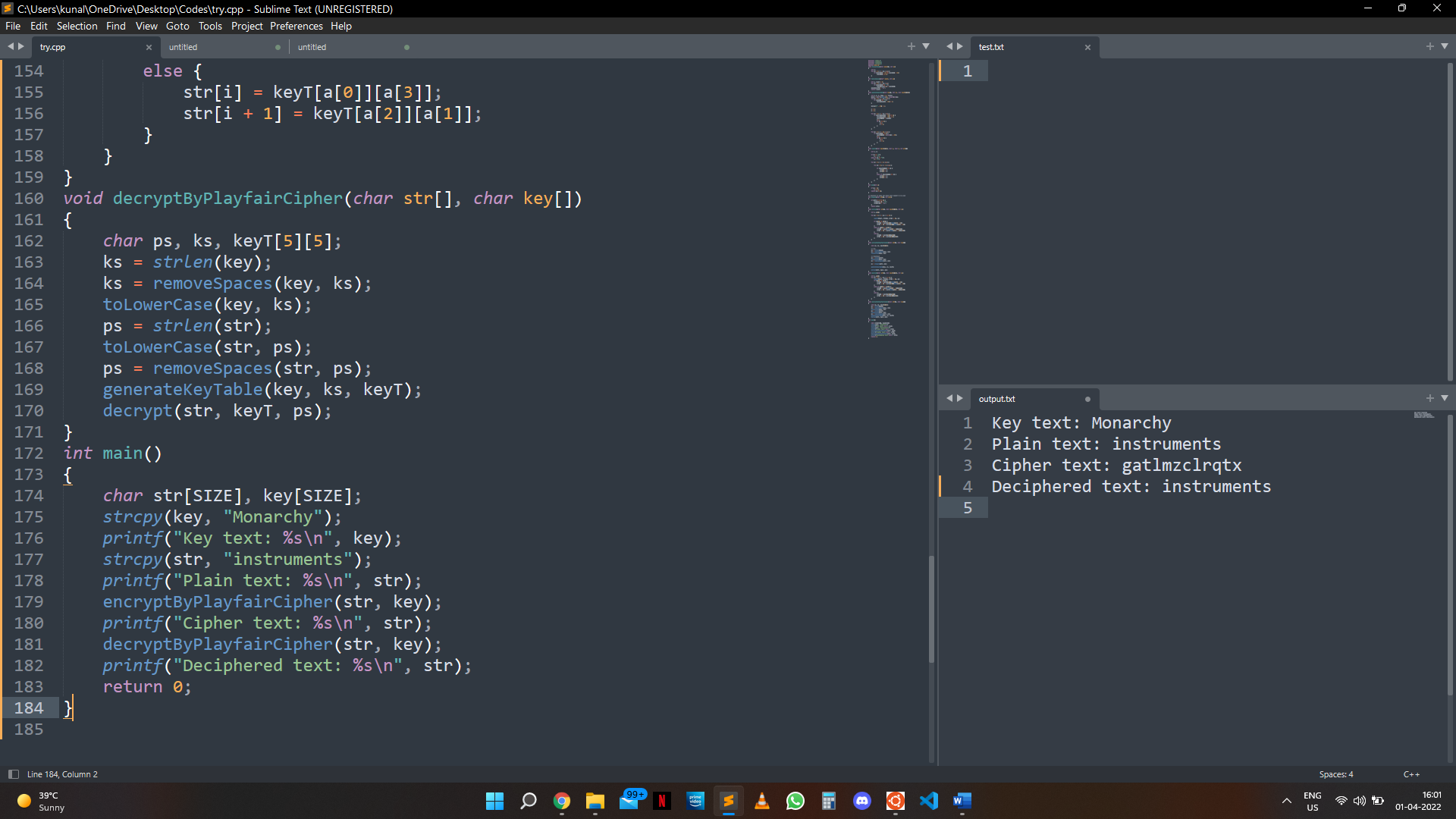
printf("Cipher text: %s\n", str);

decryptByPlayfairCipher(str, key);

printf("Deciphered text: %s\n", str);

return 0;

}

**Output:**