VIT
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

BCSE309P

Cryptography and Network Security

Assignment – 1A

Mono Alphabetic Substitution Cipher,
PlayFair Cipher,
Relative Frequencies

Name: *Yashvanth Karunakaran*
Register number: *23BAI1589*
Faculty Name: *Dr. Radhika Selvamani*
Slot: *C1+TC1*

Table of Contents

1. Introduction

This Assignment Report is to analyze the classical cryptographic techniques and independent relative frequencies of text.

Classical Cryptographic Techniques:

- Permuted Mono alphabetic substitution cipher
  Each character in the plaintext is randomly replaced by exactly one other character. This mapping is fixed for the entire message. For English alphabets 26! Mappings possible.
- PlayFair Cipher
  Encrypts two letters at a time instead of one. It hides patterns better than simple substitution because of pairs of letters encryption. It uses 5x5 or 6x6 key table filled with letters based on a keyword.

Identifying Independent Relative Frequencies:

- Monograms
- Digrams
- Trigrams

Task:

- Encryption of 100-word plain text passage using the classical cryptographic techniques and generating an encrypted text output file along with key mapping table.
- Frequency analysis of 10000-word English corpus of monogram, diagram, trigram relative frequency computation.

2. Approach Explanation:

Monoalphabetic Substitution Cipher:

Approach:

Substituting each letter in the plain text with the letter mapping in the map table and creating an encrypted substitute text.

Map Table:

Created random cipher substitution using the names by removing repeated letters: yashvanth, karunakaran, prema, zebra, wonder and then remaining letters.

Plain	Cipher	Plain	Cipher	Plain	Cipher
a	y	m	m	y	q
b	a	n	z	z	x
c	s	o	b	:	.
d	h	p	w	,	,
e	v	q	o	.	+
f	n	r	d	(space)	‘
g	t	s	c	+	:
h	k	t	i	‘	(space)
i	r	u	f		
j	u	v	j		
k	p	w	g		
l	e	x	l		

PlayFair Cipher:

Map Table (5x5 Grid):

Keyword: Yashvanth

Approach:

Splitting the text into digrams (pair of letters), if repeated letters in a digram then separate them with filler letter (x), if only one letter remaining to form a digram then add filler letter.

Same row: both letters are replaced with next right letter (wrap around).

Same column: both letters are replaced with next below (wrap around).

Rectangle: Replace with letters on the same row but at opposite ends.

Procedure:

Table Formation: Filling with non-repeative characters of the keyword followed by remaining alphabets

Map Table:

y	a	s	h	v
n	t	b	c	d
e	f	g	i/j	k
l	m	o	p	q
r	u	w	x	z

Example:

apple changed to ap | pl | ex, balloon changed to ba | lx | lo | on

ap | pl | ex   is encrypted as per map table as:

apple (plaintext) = hmqmir (encrypted text)

3. Assumptions:

- All input text is converted to lowercase.
- Standard 26-letter English alphabet.
- ‘i’ and ‘j’ are treated as same character to fit 5x5 matrix.
- Letter ‘x’ is used as a filler character between identical letters in a pair.
- Unlike standard ciphers that skips punctuation, this approach maps spaces and punctuations to unique characters in the cipher key.

4. Original Text:

“Yashvanth Karunakaran is my name, and I am a computer science student interested in data security. This assignment requires me to implement two historical ciphers using C++ programming. I’ll implement two classical encryption methods: the Monoalphabetic cipher and the Playfair cipher. The Monoalphabetic approach’s unique feature is that it maps every character to other character. The Playfair cipher, on the other hand, encrypts letter pairs using a key matrix. I’m also required to analyze letter frequencies from a large text file to find patterns. These exercises help me to understand the foundations of modern cryptography.”

Original Text:

https://github.com/Yashvanthk05/23BAI1589_Cryptographic/blob/main/1/Assignment_1PlainText_23BAI1589.txt

5. Encrypted Text:

Monoalphabetic substitution cipher (According to Table):

```
“gydhqymxh'uyovmyuyoym'kd'eg'myev,'ymh'k'ye'y'szebvxvo'dskvms
v'dxvhvmx'kmxvovdxvh'km'hyxy'dvsvokxg+'xhkd'yddktmevmx'ovw
vkovd'ev'xz'kebpvevmx'xcz'hkdxzoksyp'skbhvod'vdkmt's::'boztoyeek
```

mt+'kpp'kebpvevmx'xcz'spyddksyp'vmsogbxkzm'evxhzhd.'xhv'ezmzypbhyavxks'skbhvo'ymh'xhv'bpygnyko'skbhvo+'xhv'ezmzypbhyavxks'ybbozyshd'vmkwvv'nvyxvov'kd'xhyx'kx'eybd'vqvog'shyoysxvo'xz'zxhvo'shyoysxvo+'xhv'bpygnyko'skbhvo,'zm'xhv'zxhvo'hymh,'vmsogbxd'pvxxvo'bykod'vdkmt'y'uvg'eyxokl+'ke'ypdz'ovwvkovh'xz'ymypgfv'pvxxvo'novwvvmskvd'noze'y'pyotv'xvlx'nkpv'xz'nkmh'byxxvomd+'xhvdv'vlvoskdvd'hvpb'ev'xz'vmhvodxymh'xhv'nzvmhyxkzmd'zn'ezhvom'sogbxztoybhg+”

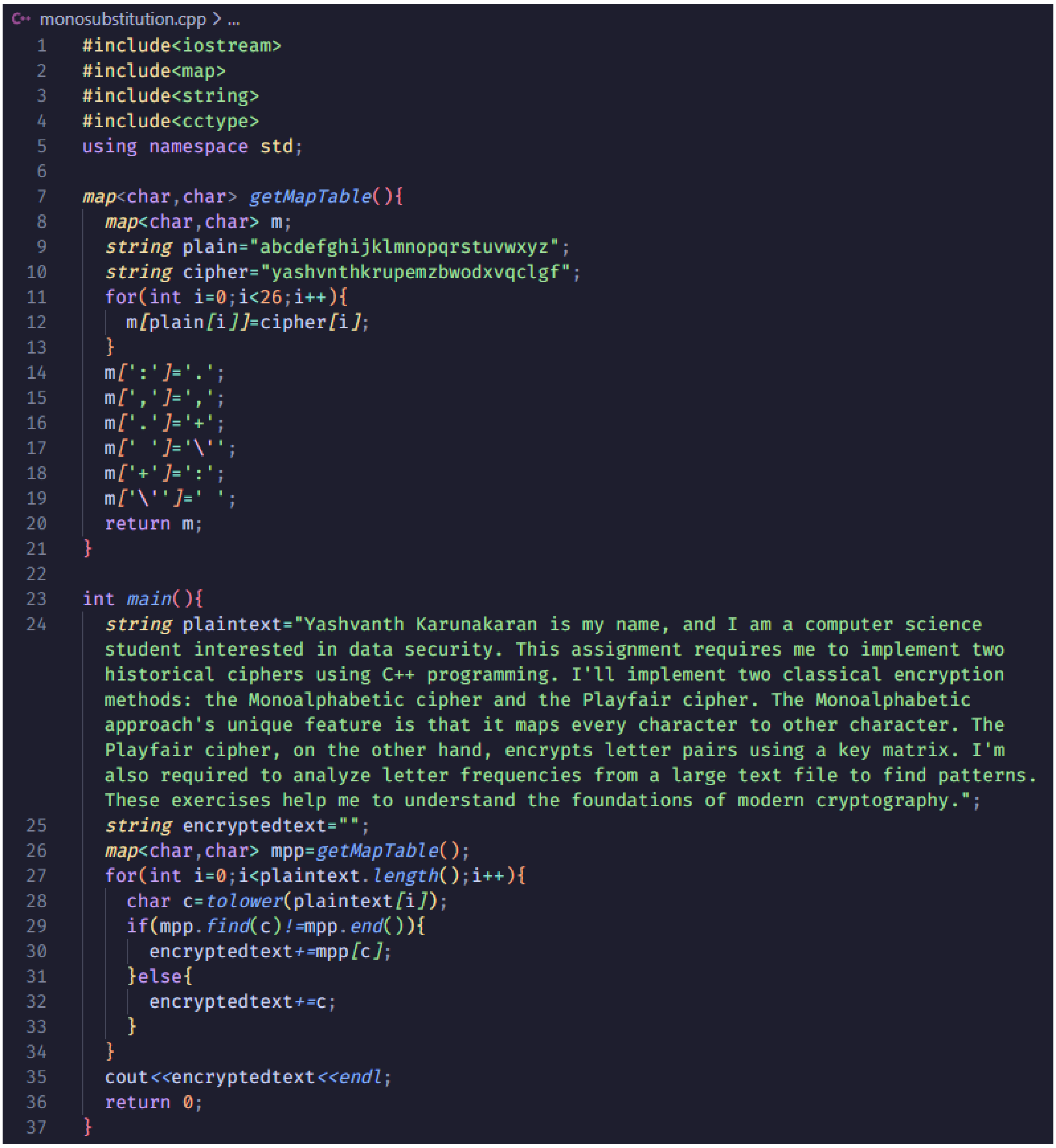
Encrypted File:

https://github.com/Yashvanthk05/23BAI1589_Cryptographic/blob/main/1/Assignment_1Data_monosubstitution_23BAI1589.txt

Map Table:

Plain	Cipher	Plain	Cipher	Plain	Cipher
a	y	m	m	y	q
b	a	n	z	z	x
c	s	o	b	:	.
d	h	p	w	,	,
e	v	q	o	.	+
f	n	r	d	(space)	‘
g	t	s	c	+	:
h	k	t	i	‘	(space)
i	r	u	f		
j	u	v	j		
k	p	w	g		
l	e	x	l		

Code:



```cpp
#include<iostream>
#include<map>
#include<string>
#include<cctype>
using namespace std;

map<char,char> getMapTable(){
  map<char,char> m;
  string plain="abcdefghijklmnopqrstuvwxyz";
  string cipher="yashvnthkrupemzbwodxvqclgf";
  for(int i=0;i<26;i++){
    m[plain[i]]=cipher[i];
  }
  m[':']='.';
  m[',']=',';
  m['.']='+';
  m[' ']='\'';
  m['+']=':';
  m['\'']=' ';
  return m;
}

int main(){
  string plaintext="Yashvanth Karunakaran is my name, and I am a computer science student interested in data security. This assignment requires me to implement two historical ciphers using C++ programming. I'll implement two classical encryption methods: the Monoalphabetic cipher and the Playfair cipher. The Monoalphabetic approach's unique feature is that it maps every character to other character. The Playfair cipher, on the other hand, encrypts letter pairs using a key matrix. I'm also required to analyze letter frequencies from a large text file to find patterns. These exercises help me to understand the foundations of modern cryptography.";
  string encryptedtext="";
  map<char,char> mpp=getMapTable();
  for(int i=0;i<plaintext.length();i++){
    char c=tolower(plaintext[i]);
    if(mpp.find(c)!=mpp.end()){
      encryptedtext+=mpp[c];
    }else{
      encryptedtext+=c;
    }
  }
  cout<<encryptedtext<<endl;
  return 0;
}
```

Source Code:

https://github.com/Yashvanthk05/23BAI1589_Cryptographic/blob/main/1/monosubstitution.cpp

Playfair cipher:

“ashvystbviyurtvfyuytghlatylfytcktuhtpomxnfwyipleniabztlecftblygynfcktntfshinwufcancphshwhgeblftbylmzexgylfbmfpqmfllecubupsghbmxethpnpxyiwywaeciblxwouypupfbeeppeoqrllftbbupbmyhwhgthrltdynmcgptlfnspbvcafllbmsmqvsngcfihipxclyytnbyiqmsamtexipxclycafllbmsmqvsngcfthxhlxmsicawcemzfgfyfaylghcatffcutohkylyhnvsuydblybmmbyixnvsuydblycailmyaehfxnpxyiwltbyimbyixyytnktdynmcyofnnfxlhfwywaecfsefaltfxehputoylwklxfylnbmstyrnrkrlcunfueylmzleipgyeupoymyuifnfucgkrlbmgktnmhcunfyeabyiygirlyipyghvlrqofnmwtnlyabytnbyigmrttvcflbbwmuqblytdynmcwouyxchr”

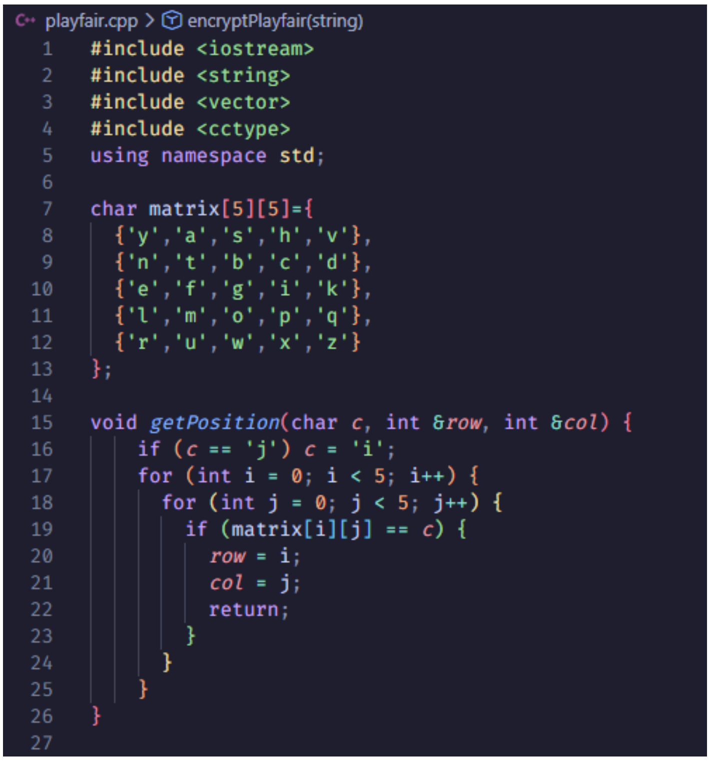
Encrypted File:

https://github.com/Yashvanthk05/23BAI1589_Cryptographic/blob/main/1/Assignment_1Data_Playfair_23BAI1589.txt

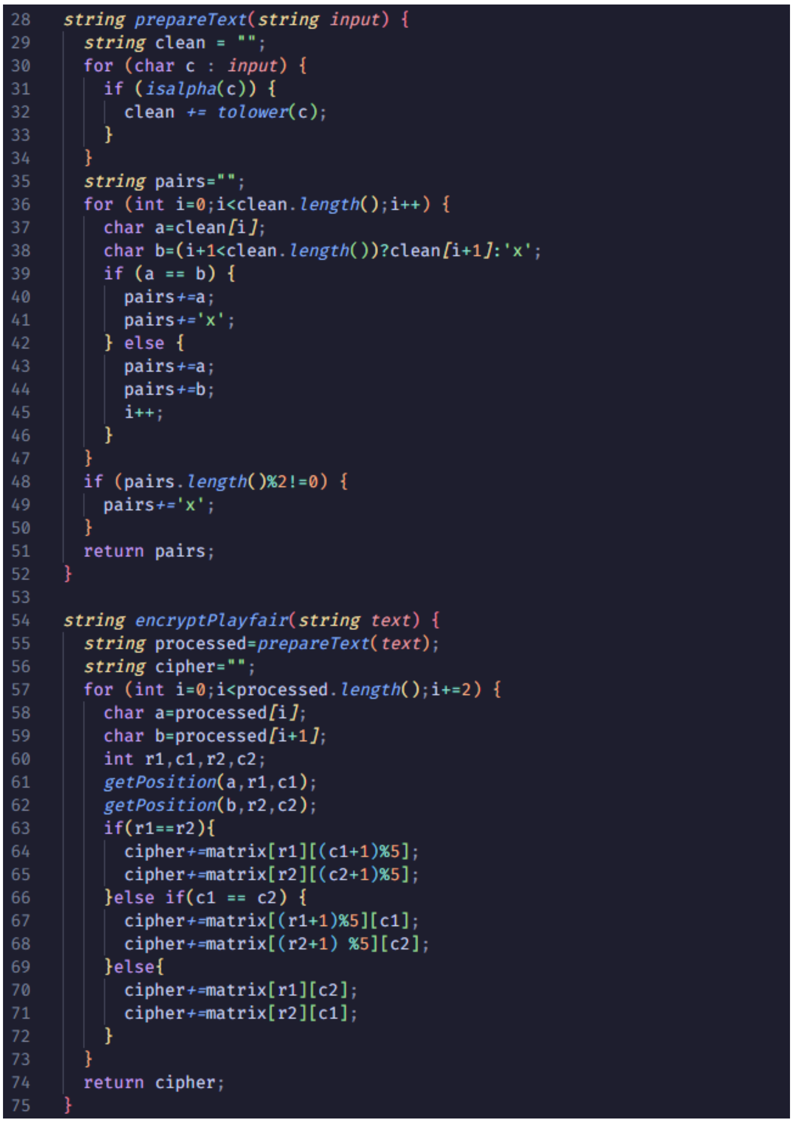
Map Table:

y	a	s	h	v
n	t	b	c	d
e	f	g	i/j	k
l	m	o	p	q
r	u	w	x	z

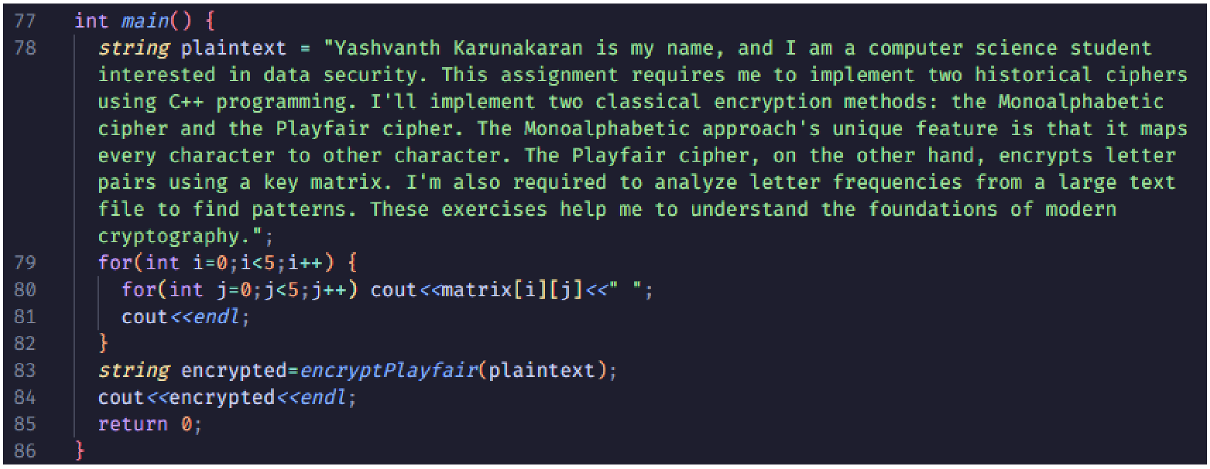
Code:



```cpp
#include <iostream>
#include <string>
#include <vector>
#include <cctype>
using namespace std;

char matrix[5][5]={
  {'y','a','s','h','v'},
  {'n','t','b','c','d'},
  {'e','f','g','i','k'},
  {'l','m','o','p','q'},
  {'r','u','w','x','z'}
};

void getPosition(char c, int &row, int &col) {
    if (c == 'j') c = 'i';
    for (int i = 0; i < 5; i++) {
      for (int j = 0; j < 5; j++) {
        if (matrix[i][j] == c) {
          row = i;
          col = j;
          return;
        }
      }
    }
}

```



```cpp
string prepareText(string input) {
    string clean = "";
    for (char c : input) {
        if (isalpha(c)) {
            clean += tolower(c);
        }
    }
    string pairs="";
    for (int i=0;i<clean.length();i++) {
        char a=clean[i];
        char b=(i+1<clean.length())?clean[i+1]:'x';
        if (a == b) {
            pairs+=a;
            pairs+='x';
        } else {
            pairs+=a;
            pairs+=b;
            i++;
        }
    }
    if (pairs.length()%2!=0) {
        pairs+='x';
    }
    return pairs;
}

string encryptPlayfair(string text) {
    string processed=prepareText(text);
    string cipher="";
    for (int i=0;i<processed.length();i+=2) {
        char a=processed[i];
        char b=processed[i+1];
        int r1,c1,r2,c2;
        getPosition(a,r1,c1);
        getPosition(b,r2,c2);
        if(r1==r2){
            cipher+=matrix[r1][(c1+1)%5];
            cipher+=matrix[r2][(c2+1)%5];
        }else if(c1 == c2) {
            cipher+=matrix[(r1+1)%5][c1];
            cipher+=matrix[(r2+1) %5][c2];
        }else{
            cipher+=matrix[r1][c2];
            cipher+=matrix[r2][c1];
        }
    }
    return cipher;
}
```



```cpp
int main() {
    string plaintext = "Yashvanth Karunakaran is my name, and I am a computer science student interested in data security. This assignment requires me to implement two historical ciphers using C++ programming. I'll implement two classical encryption methods: the Monoalphabetic cipher and the Playfair cipher. The Monoalphabetic approach's unique feature is that it maps every character to other character. The Playfair cipher, on the other hand, encrypts letter pairs using a key matrix. I'm also required to analyze letter frequencies from a large text file to find patterns. These exercises help me to understand the foundations of modern cryptography.";
    for(int i=0;i<5;i++) {
        for(int j=0;j<5;j++) cout<<matrix[i][j]<<" ";
        cout<<endl;
    }
    string encrypted=encryptPlayfair(plaintext);
    cout<<encrypted<<endl;
    return 0;
}
```

Source Code:

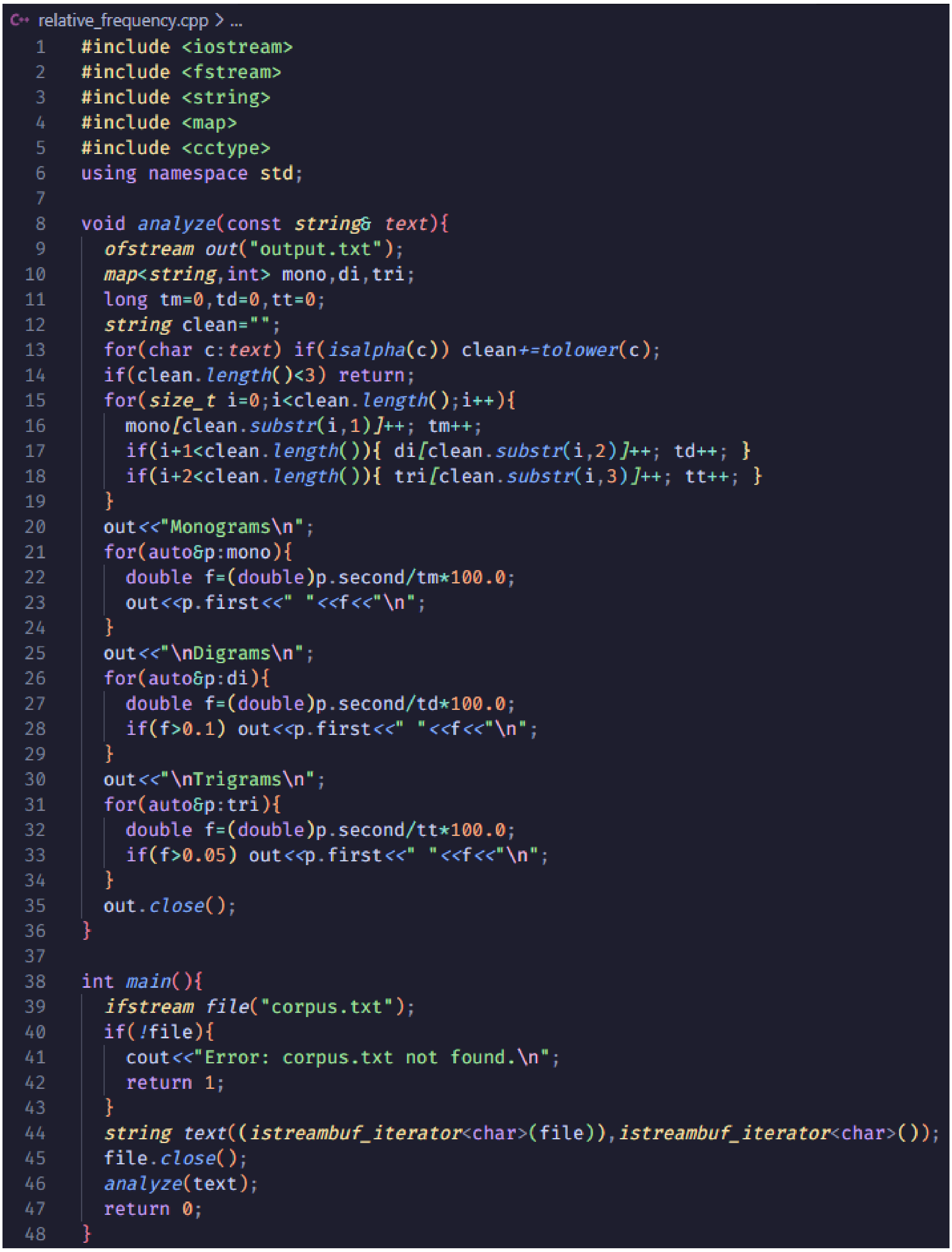
https://github.com/Yashvanthk05/23BAI1589_Cryptographic/blob/main/1/playfair.cpp

6. Independent Relative Frequency

Source File: https://www.gutenberg.org/cache/epub/35/pg35.txt

Independent relative Frequency:

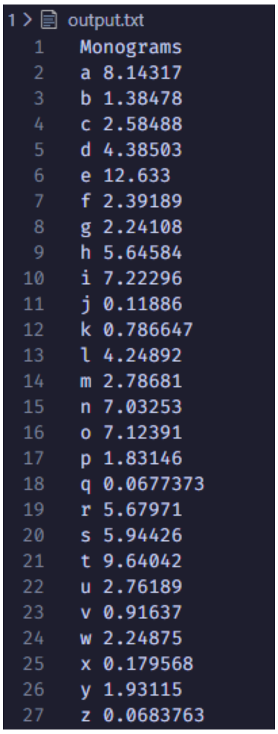
Code:



```cpp
#include <iostream>
#include <fstream>
#include <string>
#include <map>
#include <cctype>
using namespace std;

void analyze(const string& text){
    ofstream out("output.txt");
    map<string,int> mono,di,tri;
    long tm=0,td=0,tt=0;
    string clean="";
    for(char c:text) if(isalpha(c)) clean+=tolower(c);
    if(clean.length()<3) return;
    for(size_t i=0;i<clean.length();i++){
        mono[clean.substr(i,1)]++; tm++;
        if(i+1<clean.length()){ di[clean.substr(i,2)]++; td++; }
        if(i+2<clean.length()){ tri[clean.substr(i,3)]++; tt++; }
    }
    out<<"Monograms\n";
    for(auto&p:mono){
        double f=(double)p.second/tm*100.0;
        out<<p.first<<" "<<f<<"\n";
    }
    out<<"\nDigrams\n";
    for(auto&p:di){
        double f=(double)p.second/td*100.0;
        if(f>0.1) out<<p.first<<" "<<f<<"\n";
    }
    out<<"\nTrigrams\n";
    for(auto&p:tri){
        double f=(double)p.second/tt*100.0;
        if(f>0.05) out<<p.first<<" "<<f<<"\n";
    }
    out.close();
}

int main(){
    ifstream file("corpus.txt");
    if(!file){
        cout<<"Error: corpus.txt not found.\n";
        return 1;
    }
    string text((istreambuf_iterator<char>(file)),istreambuf_iterator<char>());
    file.close();
    analyze(text);
    return 0;
}
```

Output File:

https://github.com/Yashvanthk05/23BAI1589_Cryptographic/blob/main/1/output.txt



```
Monograms
a 8.14317
b 1.38478
c 2.58488
d 4.38503
e 12.633
f 2.39189
g 2.24108
h 5.64584
i 7.22296
j 0.11886
k 0.786647
l 4.24892
m 2.78681
n 7.03253
o 7.12391
p 1.83146
q 0.0677373
r 5.67971
s 5.94426
t 9.64042
u 2.76189
v 0.91637
w 2.24875
x 0.179568
y 1.93115
z 0.0683763
```

There are over 100+ Digrams and Trigrams in the source text file so I have pasted the output file GitHub link