Observation:

There are a few words have more than one appearances in the cipher text.
And repeatedly, so these words could be connective words or pronouns or verbs or prepositions that are usually crucial to the constructions of all sentences. Such as “is”, “am”, “are”, “He”, “She”, “It”, “they”, “the”, “a”, “and”, “or”, “at”, “in”, “to”, etc.

1. BPR (6 appearances in the first paragraph; 7 appearances in the second paragraph, 13 appearances in total)
2. JX (6 appearances in the first paragraph; 3 appearances in the first paragraph; 9 appearances in total)
3. BJ (2 appearances in the first paragraph; 4 appearances in the second paragraph, 6 appearances in total)
4. W (1 appearances in the first paragraph; 2 appearances in the second paragraph, 3 appearances in total) ***only “A” and “I” are one letter words in English language.

Frequencies analysis:

Frequency of the letters in this ciphertext:
A: 5
Frequency rate: 0.00773994
B: 68
Frequency rate: 0.105263
C: 5
Frequency rate: 0.00773994
D: 23
Frequency rate: 0.0356037
E: 5
Frequency rate: 0.00773994
F: 1
Frequency rate: 0.00154799
G: 1
Frequency rate: 0.00154799
H: 23
Frequency rate: 0.0356037
I: 41
Frequency rate: 0.0634675
J: 48
Frequency rate: 0.0743034
K: 49

Frequency rate: 0.0758514
L: 8
Frequency rate: 0.0123839
M: 62
Frequency rate: 0.0959752
N: 17
Frequency rate: 0.0263158
O: 7
Frequency rate: 0.0108359
P: 30
Frequency rate: 0.0464396
Q: 7
Frequency rate: 0.0108359
R: 84
Frequency rate: 0.130031
S: 17
Frequency rate: 0.0263158
T: 13
Frequency rate: 0.0201238
U: 24
Frequency rate: 0.0371517
V: 22
Frequency rate: 0.0340557
W: 47
Frequency rate: 0.0727554
X: 20
Frequency rate: 0.0309598
Y: 19
Frequency rate: 0.0294118
Z: 0
Frequency rate: 0
Total Characters: 646

Primary decryption:

For number one (BPR), my guess is “ARE” or “THE”, since the frequency of the letter ‘R’ appeared the most (84 times) in the cipher text, so it is pretty safe to guess the letter ‘R’ represents ‘E’ in plaintext. Appearance of letter ‘B’ is the second highest, and according to the relative letter frequencies of the English Language table given in the pdf, the letter ’T’ is the second highest, and

‘A’ is the third highest. Which means the word “BPR” could be very likely be “T_E” or “A_E”. Which eliminate a lot of words and made the whole process a lot easier.

Words possible (Starts with T): TAE, TEE, TIE, TOE, TYE, **THE**
Words possible (Starts with A): ABE, ADE, AKE, AME, AOE, AUE, AWE, AYE, AYE, ACE, AGE, ALE, ANE, APE, ATE, AVE, AXE, AZE, **ARE**

I bolded the words that I think that are more likely to be the actual word in plaintext like I explained, this word BPR appeared in the cipher text a lot so it should be an important component in the text. There isn’t much reason for words like ape or toe be in the text a lot unless the topic of the text is about this object, which is still not likely for someone to mention an object over and over again, when they can just use “IT” to refer to the object after the first appearance.

The words that I underlined are mostly verbs, which I think it might fit.

I think it is more possible to be “THE”.

Similarly, I picked “OF” and “TO” for cipher “JX”. But I eliminated the possibility of “TO” since BJ could be “TO”, and there’s common letter of “BJ”, “JX”, and “BPR”. So we can almost confirm that “B” is T, then “J” cannot be T, if it is “O”, all the guesses will be logical. W is a single letter word, and the only single letter words are a and I, so I picked both, this needed further substitution for confirm which one is correct.

Methods

First, I observe the words with higher frequencies, these words with high frequencies are easier to identify, as mentioned in the first part of the report. Second, I used the programme to calculate the frequencies of the letters. After that, I used two methods to decrypt the cipher.

The first method is to manually substitute the words according to grammatical structure of English Language as mentioned in the primary decryption part. Then I picked the words with a few letters guessed, and like filling in the blanks, according to the word length and the letters figured, I substitute letter that would make the cipher word a word. For example, “bprjuwri”, with BPR, J, and W (unconfirmed), the word would be “THEO_A_ _” or “THEO_I_ _”. This single word helps to decrypt 8 letters which is almost 1/3 of the key, so this is very efficient and more accurate than method two. My guess is “THEORIES”. I then use the search and replace function to change all the letter I found in this words and replace it with the letter I think was the key. And with this information and method, I continue to guess all the words in the text, it took me about 15 minutes to finish.

The below is the decrypted message I got using method 1:

BECAUSE THE PRACTICE OF THE BASIC MOVEMENTS OF KATA IS THE FOCUS AND MASTERY OF SELF IS THE ESSENCE OF MATSUBAYASHI RYU KARATE DO I SHALL TRY TO ELUCIDATE THE MOVEMENTS OF THE KATA ACCORDING TO MY INTERPRETATION BASED ON FORTY YEARS OF STUDY

IT IS NOT AN EASY TASK TO EXPLAIN EACH MOVEMENT AND ITS SIGNIFICANCE AND SOME MUST REMAIN UNEXPLAINED TO GIVE A COMPLETE EXPLANATION ONE WOULD HAVE TO BE QUALIFIED AND INSPIRED TO SUCH AN EXTENT THAT HE COULD REACH THE STATE OF ENLIGHTENED MIND CAPABLE OF RECOGNIZING SOUNDLESS SOUND AND SHAPELESS SHAPE I DO NOT DEEM MYSELF THE FINAL AUTHORITY BUT MY EXPERIENCE WITH KATA HAS LEFT NO DOUBT THAT THE FOLLOWING IS THE PROPER APPLICATION AND INTERPRETATION I OFFER MY THEORIES IN THE HOPE THAT THE ESSENCE OF OKINAWAN KARATE WILL REMAIN INTACT

These are the words I decrypted in the order I decrypted them.

Capital letter represents the letter I have decrypted, lowercase represents the letter in cipher text.

1. bpr > THE : b=T; p=H; r=R
2. jx > OF : j=O; x=F
3. w > I : w=I
4. THEOuIEi > THEORIES : u=R; i=S ; w=I (confirms w=I)
5. kOT > NOT : k=N
6. mN =AN : m=A
7. EASt > EASY : t=Y
8. TASq > TASK : q=K
9. AnTHORITY >AUTHORITY : n=U
10. ESSENvE > ESSENCE : v=C
11. lASIS > BASIS : l=B
12. yASTERY > MASTERY : y=M
13. SEhF >SELF : h=L
14. cILL > WILL : c=W
15. sRACTICE > PRACTICE : s=P
16. ACCORdINo > ACCORDING : d=D; o=G

17. EaPLAIN > EXPLAIN : a=X
18. HAeE > HAVE : e=V
19. RECOGNIgING > RECOGNIZING : g=Z
20. fUALIFIED > QUALIFIED : f=Q

In total 20 guesses, I decrypted the entire message, with the key of 26 letters.

The second method is simple. Match the cipher letter frequencies with the English Language Letter Frequencies.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	A	5			R	84		E	0.127					A	0.0817
2	B	68			B	68		T	0.0906					B	0.015
3	C	5			M	62		A	0.0817					C	0.0278
4	D	23			K	49		O	0.0751					D	0.0425
5	E	5			J	48		I	0.0697					E	0.127
6	F	1			W	47		N	0.0675					F	0.0223
7	G	1			I	41		S	0.0633					G	0.0202
8	H	23			P	30		H	0.0609					H	0.0609
9	I	41			U	24		R	0.0599					I	0.0697
10	J	48			D	23		D	0.0425					J	0.0015
11	K	49			H	23		L	0.0403					K	0.0077
12	L	8			V	22		C	0.0278					L	0.0403
13	M	62			X	20		U	0.0276					M	0.0241
14	N	17			Y	19		M	0.0241					N	0.0675
15	O	7			S	17		W	0.0236					O	0.0751
16	P	30			N	17		F	0.0223					P	0.0193
17	Q	7			T	13		G	0.0202					Q	0.001
18	R	84			L	8		Y	0.0197					R	0.0599
19	S	17			O	7		P	0.0193					S	0.0633
20	T	13			Q	7		B	0.015					T	0.0906
21	U	24			A	5		V	0.0098					U	0.0276
22	V	22			C	5		K	0.0077					V	0.0098
23	W	47			E	5		J	0.0015					W	0.0236
24	X	20			F	1		X	0.0015					X	0.0015
25	Y	19			G	1		Q	0.001					Y	0.0197
26	Z	0			Z	0		Z	0.0007					Z	0.0007

The E column and H column is the answer, E Column is the cipher text letters, and its key is on H column in the same row. I did it on two different software to compare how much time I spent. It takes 2 minutes to finish on excel, 18 minute to make a programme using C++ language. For decrypting one substitution cipher text, using excel would be great, very efficient. But building a programme is excellent for long term. I can use the same program to decrypt many substitution cipher text with 1-4 minutes depending on the text size, I only have to insert new text file and that could be done. I get many different results, since there are letters having the same frequencies. Also, K, J, W, X, T, L, Z from the cipher text interpreted just by letter frequencies are wrong. For the letters with same frequencies, arranging the order randomly and match with the English letter frequency table, 9 out 11 of them are wrong, the only correct ones are D and H from the cipher text. With only using this method, without any supplemented method, the accuracy is 38% (only 10 letters are decrypted correctly).

Method Analysis and Solutions

From the first method, I learn that it would be harder to decrypt with a short cipher text since it could have many possibilities. But with longer text, it is way easier to decrypt since the word will appear more often, and it is harder to analyse the letter frequencies without sufficient number of letter samples. Which means this method would be pretty safe if it is used in a single word or sentence with no letter repeating in the same word, there is no much sample for analysing the word frequency. For example, the cipher text is just bpr, it could be almost all three letter words, there are really that many combination, according to the scrabble player's dictionary, there are only 1,065 three letter words, and eliminating the ones with same letter in a word, such as wow, there are not a lot. But this would be relatively harder to decrypt than having a longer passage. This method of encryption would be more effective against attack if it is for 8 letter words with little to no same letter in the same word, since there are about 80000 words (According to the Scrabble player's dictionary). This is so far I think the best length for one word with no repeating or very little repeating letter. Even longer length words have no repeating letter words or low repeating rate words logically have more difficulties to guess, but there are three factors why I think 6-8 letters word with no repeating letters are better. First, there are fewer words with the length longer than 8 words. Second, if there is words with long length and no repeating word, which will be relatively unique and easier to crack. Third, if there are repeating letters, which makes it easier for the attacker to figure out the word, with the information of where the repeating letters locate in the word, and it would also be unique, which will make the word range of guessing the cipher way smaller.

From the second method, I highlighted 11 rows of data, representing 11 letters from the cipher text. This method needs more complements and assistance to get higher accuracy, but it is relatively efficient than method one.

Therefore, I used two methods to finish this decryption, first one to find out the highest frequency letters "E", "T", "A", since these three letters have distinctively high appearance frequencies in both the cipher text and the English language letter frequency table, which means these are more likely to be correct. After that, check manually if other letters are matches to create a word with meaning.

Key

Row 1 is Plain Text, Row 2 is letters from Cipher Text

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
M	L	V	D	R	X	O	P	W	Z	Q	H	Y	K	J	S	F	U	I	B	N	E	C	A	T	G

Code:

main.cpp:

```
#include <iostream>
#include <fstream>
#include <numeric>
#include <vector>
#include <iterator>
#include <map>
using namespace std;

int main() {
  multimap<float,char> list;//letter and freq
  multimap<float,char> list2;//letter and distribution
  multimap<float, char> list3; //letter and distribution from table
  multimap<char,char> list4; //conversion table
  vector<char> letter {'a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','w','x','y','z'};
    vector<float> count (26,0);
    vector<float> distribution (26,0);
    vector<float>
distribution_table{0.0817,0.0150,0.0278,0.0425,0.1270,0.0223,0.0202,0.0609,0.0697,0.0015,0.007
7,0.0403,0.0241,0.0675,0.0751,0.0193,0.0010,0.0599,0.0633,0.0906,0.0276,0.0098,0.0236,0.0015,
0.0197,0.0007};

    fstream file;
    string word, filename;

    filename = "ciphertext.txt";

    file.open(filename.c_str());

    while (file >> word)
    {
       for(int i=0;i<word.length();i++){

         for(int j=0;j<26;j++){
           if(word.at(i)!=letter.at(j)){
             continue;
           }
```

```
        else if(word.at(i)==letter.at(j)){
          count.at(j)++;
        }
       }
     }

}

float sum = accumulate(count.begin(),count.end(),0);

for(int y=0;y<26;y++){
  list.insert(pair<float,char>((count.at(y)),letter.at(y)));
  distribution.at(y)=count.at(y)/sum;
  list2.insert(pair<float,char>((distribution.at(y)),letter.at(y)));
  list3.insert(pair<float,char>((distribution_table.at(y)),letter.at(y)));
}


map<float,char>:: iterator it;
for (it=list.begin() ; it!=list.end() ; it++){
    cout << "Letter: " << (*it).second << " ";
    cout <<"Frequency: " << (*it).first << endl;}

cout<<endl<<endl<<endl;

for (it=list2.begin() ; it!=list2.end() ; it++){
    cout << "Letter: " << (*it).second << " ";
    cout <<"Distribution: " << (*it).first << endl;}

    cout<<"Total character: "<<sum<<endl;

cout<<endl<<endl<<endl;

for (it=list3.begin() ; it!=list3.end() ; it++){
    cout << "Letter From Table: " << (*it).second << " ";
    cout <<"Frequency From Table: " << (*it).first << endl;}

    string text = "lrvmnir bpr sumvbwvr jx bpr lmiwv yjeryrkbi jx qmbm wi bpr xjvni mkd
ymibrut jx irhx wi bpr riirkvr jx ymbinlmtmipw utn qmumbr dj w ipmhh but bj rhnvwdmbr bpr
yjeryrkbi jx bpr qmbm mvvjudwko bj yt wkbrusurbmbwjk lmird jk xjubt trmui jx ibndt";
```

string text2 ="wb wi kjb mk rmit bmiq bj rashmwk rmvp yjeryrkb mkd wbi iwokwxwvmkvr mkd ijyr ynib urymwk nkrashmwkrd bj ower m vjyshrbr rashmkmbwjk jkr cjnhd pmer bj lr fnmhwxwrd mkd wkiswurd bj invp mk rabrkb bpmb pr vjnhd urmvp bpr ibmbr jx rkhwopbrkrd ywkd vmsmlhr jx urvjokwgwko ijnkdhrii ijnkd mkd ipmsrhrii ipmsr w dj kjb drry ytirhx bpr xwkmh mnbpjuwbt lnb yt rasruwrkvr cwbp qmbm pmi hrxb kj djnlb bpmb bpr xjhhjcwko wi bpr sujsru msshwvmbwjk mkd wkbrusurbmbwjk w jxxru yt bprjuwri wk bpr pjsr bpmb bpr riirkvr jx jqwkmcmk qmumbr cwhh urymwk wkbmvb";

```
cout<<"\n\ntranslation: "<<endl<<endl;

for (int t=0; t<text.length();t++){
 if(text.at(t)=='z'){
  cout << "z";
 }
 else if(text.at(t)=='f'){
  cout << "q";
 }
 else if(text.at(t)=='g'){
  cout << "j";
 }
 else if(text.at(t)=='a'){
  cout << "x";
 }
 else if(text.at(t)=='c'){
  cout << "k";
 }
 else if(text.at(t)=='e'){
  cout << "v";
 }
 else if(text.at(t)=='o'){
  cout << "b";
 }
 else if(text.at(t)=='q'){
  cout << "p";
 }
 else if(text.at(t)=='l'){
  cout << "y";
 }
 else if(text.at(t)=='t'){
```

```
  cout << "g";
}
else if(text.at(t)=='n'){
  cout << "f";
}
else if(text.at(t)=='s'){
  cout << "w";
}
else if(text.at(t)=='y'){
  cout << "m";
}
else if(text.at(t)=='x'){
  cout << "u";
}
else if(text.at(t)=='v'){
  cout << "c";
}
else if(text.at(t)=='d'){
  cout << "l";
}
else if(text.at(t)=='h'){
  cout << "d";
}
else if(text.at(t)=='u'){
  cout << "r";
}
else if(text.at(t)=='p'){
  cout << "h";
}
else if(text.at(t)=='i'){
  cout << "s";
}
else if(text.at(t)=='w'){
  cout << "n";
}
else if(text.at(t)=='j'){
  cout << "i";
}
else if(text.at(t)=='k'){
  cout << "o";
```

```
  }
  else if(text.at(t)=='m'){
    cout << "a";
  }
  else if(text.at(t)=='b'){
    cout << "t";
  }
  else if(text.at(t)=='r'){
    cout << "e";
  }
  else if(text.at(t)==' '){
    cout << " ";
  }
}

cout<<endl<<endl;

for (int n=0; n<text2.length();n++){

  if(text2.at(n)=='z'){
    cout << "z";
  }
  else if(text2.at(n)=='f'){
    cout << "q";
  }
  else if(text2.at(n)=='g'){
    cout << "j";
  }
  else if(text2.at(n)=='a'){
    cout << "x";
  }
  else if(text2.at(n)=='c'){
    cout << "k";
  }
  else if(text2.at(n)=='e'){
    cout << "v";
  }
  else if(text2.at(n)=='o'){
    cout << "b";
  }
```

```
else if(text2.at(n)=='q'){
  cout << "p";
}
else if(text2.at(n)=='l'){
  cout << "y";
}
else if(text2.at(n)=='t'){
  cout << "g";
}
else if(text2.at(n)=='n'){
  cout << "f";
}
else if(text2.at(n)=='s'){
  cout << "w";
}
else if(text2.at(n)=='y'){
  cout << "m";
}
else if(text2.at(n)=='x'){
  cout << "u";
}
else if(text2.at(n)=='v'){
  cout << "c";
}
else if(text2.at(n)=='d'){
  cout << "l";
}
else if(text2.at(n)=='h'){
  cout << "d";
}
else if(text2.at(n)=='u'){
  cout << "r";
}
else if(text2.at(n)=='p'){
  cout << "h";
}
else if(text2.at(n)=='i'){
  cout << "s";
}
else if(text2.at(n)=='w'){
```

```
        cout << "n";
      }
      else if(text2.at(n)=='j'){
        cout << "i";
      }
      else if(text2.at(n)=='k'){
        cout << "o";
      }
      else if(text2.at(n)=='m'){
        cout << "a";
      }
      else if(text2.at(n)=='b'){
        cout << "t";
      }
      else if(text2.at(n)=='r'){
        cout << "e";
      }
      else if(text2.at(n)==' '){
        cout << " ";
      }
    }

  return 0;
}
```

ciphertext.txt:

lrvmnir bpr sumvbwvr jx bpr lmiwv yjeryrkbi jx qmbm wi bpr xjvni mkd ymibrut jx irhx wi bpr riirkvr jx ymbinlmtmipw utn qmumbr dj w ipmhh but bj rhnvwdmbr bpr yjeryrkbi jx bpr qmbm mvvjudwko bj yt wkbrusurbmbwjk lmird jk xjubt trmui jx ibndt

wb wi kjb mk rmit bmiq bj rashmwk rmvp yjeryrkb mkd wbi iwokwxwvmkvr mkd ijyr ynib urymwk nkrashmwkrd bj ower m vjyshrbr rashmkmbwjk jkr cjnhd pmer bj lr fnmhwxwrd mkd wkiswurd bj invp mk rabrkb bpmb pr vjnhd urmvp bpr ibmbr jx rkhwopbrkrd ywkd vmsmlhr jx urvjokwgwko ijnkdhrii ijnkd mkd ipmsrhrii ipmsr w dj kjb drry ytirhx bpr xwkmh mnbpjuwbt lnb yt rasruwrkvr cwbp qmbm pmi hrxb kj djnlb bpmb bpr xjhhjcwko wi bpr sujsru msshwvmbwjk mkd wkbrusurbmbwjk w jxxru yt bprjuwri wk bpr pjsr bpmb bpr riirkvr jx jqwkmcmk qmumbr cwhh urymwk wkbmvb

Output:

Letter: z Frequency: 0
Letter: f Frequency: 1
Letter: g Frequency: 1
Letter: a Frequency: 5
Letter: c Frequency: 5
Letter: e Frequency: 5
Letter: o Frequency: 7
Letter: q Frequency: 7
Letter: l Frequency: 8
Letter: t Frequency: 13
Letter: n Frequency: 17
Letter: s Frequency: 17
Letter: y Frequency: 19
Letter: x Frequency: 20
Letter: v Frequency: 22
Letter: d Frequency: 23
Letter: h Frequency: 23
Letter: u Frequency: 24
Letter: p Frequency: 30
Letter: i Frequency: 41
Letter: w Frequency: 47
Letter: j Frequency: 48
Letter: k Frequency: 49
Letter: m Frequency: 62
Letter: b Frequency: 68
Letter: r Frequency: 84

Letter: z Distribution: 0
Letter: f Distribution: 0.00154799
Letter: g Distribution: 0.00154799
Letter: a Distribution: 0.00773994
Letter: c Distribution: 0.00773994
Letter: e Distribution: 0.00773994
Letter: o Distribution: 0.0108359
Letter: q Distribution: 0.0108359

Letter: l Distribution: 0.0123839
Letter: t Distribution: 0.0201238
Letter: n Distribution: 0.0263158
Letter: s Distribution: 0.0263158
Letter: y Distribution: 0.0294118
Letter: x Distribution: 0.0309598
Letter: v Distribution: 0.0340557
Letter: d Distribution: 0.0356037
Letter: h Distribution: 0.0356037
Letter: u Distribution: 0.0371517
Letter: p Distribution: 0.0464396
Letter: i Distribution: 0.0634675
Letter: w Distribution: 0.0727554
Letter: j Distribution: 0.0743034
Letter: k Distribution: 0.0758514
Letter: m Distribution: 0.0959752
Letter: b Distribution: 0.105263
Letter: r Distribution: 0.130031
Total character: 646

Letter From Table: z Frequency From Table: 0.0007
Letter From Table: q Frequency From Table: 0.001
Letter From Table: j Frequency From Table: 0.0015
Letter From Table: x Frequency From Table: 0.0015
Letter From Table: k Frequency From Table: 0.0077
Letter From Table: v Frequency From Table: 0.0098
Letter From Table: b Frequency From Table: 0.015
Letter From Table: p Frequency From Table: 0.0193
Letter From Table: y Frequency From Table: 0.0197
Letter From Table: g Frequency From Table: 0.0202
Letter From Table: f Frequency From Table: 0.0223
Letter From Table: w Frequency From Table: 0.0236
Letter From Table: m Frequency From Table: 0.0241
Letter From Table: u Frequency From Table: 0.0276
Letter From Table: c Frequency From Table: 0.0278
Letter From Table: l Frequency From Table: 0.0403
Letter From Table: d Frequency From Table: 0.0425
Letter From Table: r Frequency From Table: 0.0599

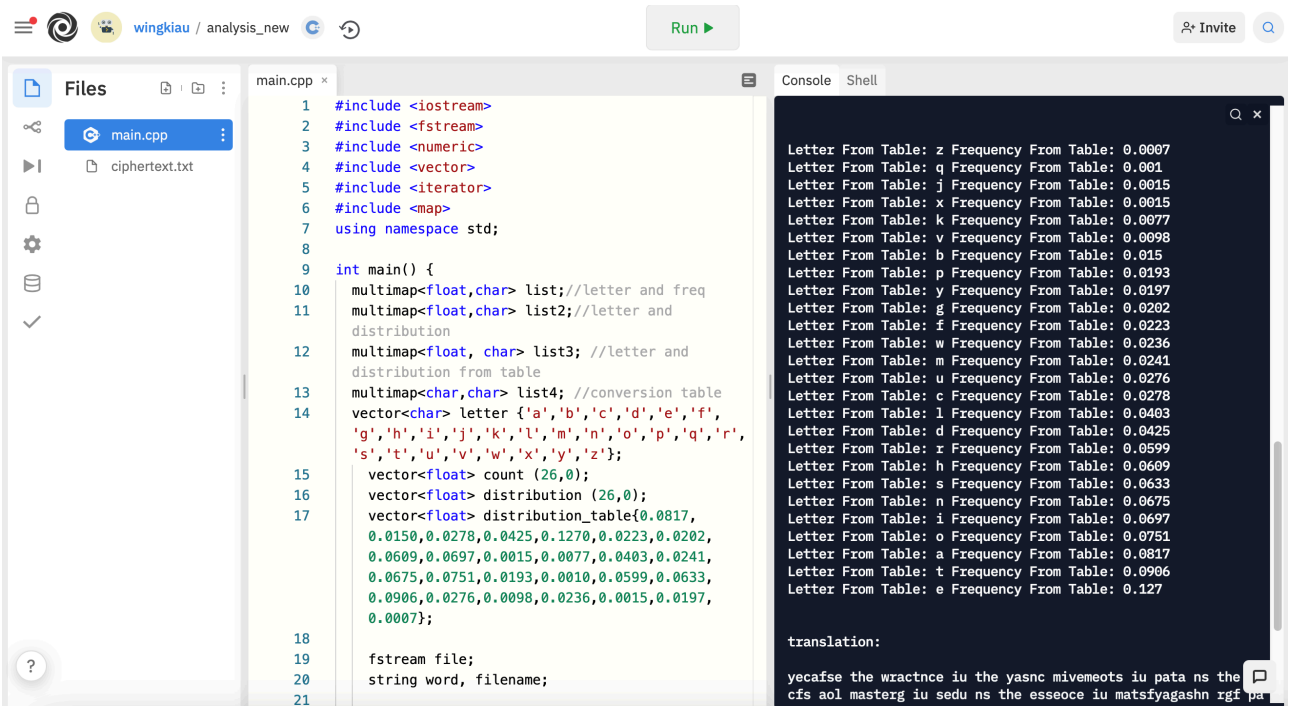
Letter From Table: h Frequency From Table: 0.0609
Letter From Table: s Frequency From Table: 0.0633
Letter From Table: n Frequency From Table: 0.0675
Letter From Table: i Frequency From Table: 0.0697
Letter From Table: o Frequency From Table: 0.0751
Letter From Table: a Frequency From Table: 0.0817
Letter From Table: t Frequency From Table: 0.0906
Letter From Table: e Frequency From Table: 0.127

translation:

yecafse the wractnce iu the yasnc mivemeots iu pata ns the uicfs aol masterg iu sedu ns the esseoce iu matsfyagashn rgf parate li n shadd trg ti edfcnlate the mivemeots iu the pata accirlnob ti mg noterwretatnio yasel io uirtg gears iu stflg

nt ns oit ao easg tasp ti exwdano each mivemeot aol nts snbonuncaoce aol sime mfst remano foexwdanoel ti bnve a cimwdete exwdaoatnio ioe kifdl have ti ye qfadnunel aol noswnrel ti sfch ao exteot that he cifdl reach the state iu eodnbhteoel mnol cawayde iu recibonjnob sifoldess sifol aol shawedess shawe n li oit leem mgsedu the unoad afthirntg yft mg exwerneoce knth pata has deut oi lifyt that the uiddiknob ns the wriwer awwdncatnio aol noterwretatnio n iuuer mg theirnes no the hiwe that the esseoce iu ipnoakao parate kndd remano notact

(Please go back for the correct translation, this translation is not accurate as I mentioned above. This translation is generated by the programme based on the letter frequencies only, without taking grammar and logic in consideration.)



wingkiau / analysis_new

Run

Invite

Files

main.cpp

ciphertext.txt

main.cpp

```cpp
#include <iostream>
#include <fstream>
#include <numeric>
#include <vector>
#include <iterator>
#include <map>
using namespace std;

int main() {
  multimap<float,char> list;//letter and freq
  multimap<float,char> list2;//letter and
  distribution
  multimap<float, char> list3; //letter and
  distribution from table
  multimap<char,char> list4; //conversion table
  vector<char> letter {'a','b','c','d','e','f',
  'g','h','i','j','k','l','m','n','o','p','q','r',
  's','t','u','v','w','x','y','z'};
    vector<float> count (26,0);
    vector<float> distribution (26,0);
    vector<float> distribution_table{0.0817,
    0.0150,0.0278,0.0425,0.1270,0.0223,0.0202,
    0.0609,0.0697,0.0015,0.0077,0.0403,0.0241,
    0.0675,0.0751,0.0193,0.0010,0.0599,0.0633,
    0.0906,0.0276,0.0098,0.0236,0.0015,0.0197,
    0.0007};

    fstream file;
    string word, filename;
```

Console Shell

```
Letter From Table: z Frequency From Table: 0.0007
Letter From Table: q Frequency From Table: 0.001
Letter From Table: j Frequency From Table: 0.0015
Letter From Table: x Frequency From Table: 0.0015
Letter From Table: k Frequency From Table: 0.0077
Letter From Table: v Frequency From Table: 0.0098
Letter From Table: b Frequency From Table: 0.015
Letter From Table: p Frequency From Table: 0.0193
Letter From Table: y Frequency From Table: 0.0197
Letter From Table: g Frequency From Table: 0.0202
Letter From Table: f Frequency From Table: 0.0223
Letter From Table: w Frequency From Table: 0.0236
Letter From Table: m Frequency From Table: 0.0241
Letter From Table: u Frequency From Table: 0.0276
Letter From Table: c Frequency From Table: 0.0278
Letter From Table: l Frequency From Table: 0.0403
Letter From Table: d Frequency From Table: 0.0425
Letter From Table: r Frequency From Table: 0.0599
Letter From Table: h Frequency From Table: 0.0609
Letter From Table: s Frequency From Table: 0.0633
Letter From Table: n Frequency From Table: 0.0675
Letter From Table: i Frequency From Table: 0.0697
Letter From Table: o Frequency From Table: 0.0751
Letter From Table: a Frequency From Table: 0.0817
Letter From Table: t Frequency From Table: 0.0906
Letter From Table: e Frequency From Table: 0.127


translation:

yecafse the wractnce iu the yasnc mivemeots iu pata ns the
cfs aol masterg iu sedu ns the esseoce iu matsfyagashn rgf pa
```

wingkiau / analysis_new

Run

Invite

Files

main.cpp

ciphertext.txt

main.cpp

```cpp
#include <iostream>
#include <fstream>
#include <numeric>
#include <vector>
#include <iterator>
#include <map>
using namespace std;

int main() {
  multimap<float,char> list;//letter and freq
  multimap<float,char> list2;//letter and
  distribution
  multimap<float, char> list3; //letter and
  distribution from table
  multimap<char,char> list4; //conversion table
  vector<char> letter {'a','b','c','d','e','f',
  'g','h','i','j','k','l','m','n','o','p','q','r',
  's','t','u','v','w','x','y','z'};
    vector<float> count (26,0);
    vector<float> distribution (26,0);
    vector<float> distribution_table{0.0817,
    0.0150,0.0278,0.0425,0.1270,0.0223,0.0202,
    0.0609,0.0697,0.0015,0.0077,0.0403,0.0241,
    0.0675,0.0751,0.0193,0.0010,0.0599,0.0633,
    0.0906,0.0276,0.0098,0.0236,0.0015,0.0197,
    0.0007};

    fstream file;
    string word, filename;
```

Console Shell

```
Letter: z Distribution: 0
Letter: f Distribution: 0.00154799
Letter: g Distribution: 0.00154799
Letter: a Distribution: 0.00773994
Letter: c Distribution: 0.00773994
Letter: e Distribution: 0.00773994
Letter: o Distribution: 0.0108359
Letter: q Distribution: 0.0108359
Letter: l Distribution: 0.0123839
Letter: t Distribution: 0.0201238
Letter: n Distribution: 0.0263158
Letter: s Distribution: 0.0263158
Letter: y Distribution: 0.0294118
Letter: x Distribution: 0.0309598
Letter: v Distribution: 0.0340557
Letter: d Distribution: 0.0356037
Letter: h Distribution: 0.0356037
Letter: u Distribution: 0.0371517
Letter: p Distribution: 0.0464396
Letter: i Distribution: 0.0634675
Letter: w Distribution: 0.0727554
Letter: j Distribution: 0.0743034
Letter: k Distribution: 0.0758514
Letter: m Distribution: 0.0959752
Letter: b Distribution: 0.105263
Letter: r Distribution: 0.130031
Total character: 646



Letter From Table: z Frequency From Table: 0.0007
Letter From Table: q Frequency From Table: 0.001
Letter From Table: j Frequency From Table: 0.0015
```

wingkiau / analysis_new

Run

Invite

Files

main.cpp

ciphertext.txt

main.cpp

```cpp
#include <iostream>
#include <fstream>
#include <numeric>
#include <vector>
#include <iterator>
#include <map>
using namespace std;

int main() {
  multimap<float,char> list;//letter and freq
  multimap<float,char> list2;//letter and
  distribution
  multimap<float, char> list3; //letter and
  distribution from table
  multimap<char,char> list4; //conversion table
  vector<char> letter {'a','b','c','d','e','f',
  'g','h','i','j','k','l','m','n','o','p','q','r',
  's','t','u','v','w','x','y','z'};
    vector<float> count (26,0);
    vector<float> distribution (26,0);
    vector<float> distribution_table{0.0817,
    0.0150,0.0278,0.0425,0.1270,0.0223,0.0202,
    0.0609,0.0697,0.0015,0.0077,0.0403,0.0241,
    0.0675,0.0751,0.0193,0.0010,0.0599,0.0633,
    0.0906,0.0276,0.0098,0.0236,0.0015,0.0197,
    0.0007};

    fstream file;
    string word, filename;
```

Console Shell

```
Letter: z Frequency: 0
Letter: f Frequency: 1
Letter: g Frequency: 1
Letter: a Frequency: 5
Letter: c Frequency: 5
Letter: e Frequency: 5
Letter: o Frequency: 7
Letter: q Frequency: 7
Letter: l Frequency: 8
Letter: t Frequency: 13
Letter: n Frequency: 17
Letter: s Frequency: 17
Letter: y Frequency: 19
Letter: x Frequency: 20
Letter: v Frequency: 22
Letter: d Frequency: 23
Letter: h Frequency: 23
Letter: u Frequency: 24
Letter: p Frequency: 30
Letter: i Frequency: 41
Letter: w Frequency: 47
Letter: j Frequency: 48
Letter: k Frequency: 49
Letter: m Frequency: 62
Letter: b Frequency: 68
Letter: r Frequency: 84



Letter: z Distribution: 0
Letter: f Distribution: 0.00154799
Letter: g Distribution: 0.00154799
Letter: a Distribution: 0.00773994
Letter: c Distribution: 0.00773994
```