HW1: Password Cracking

name: Wang Haoyuan

number: 3220105114

ciphertext 1: by caesar algorithm

the answer:

- the ciphertext is: "FBUQIUUDSHOFJOEKHDQCUMYJXJXUIQCUAUOQDTKFBEQTJEBUQHDYDWYDPZK"
- the plaintext is: "PLEASE ENCRYPT YOUR NAME WITH THE SAME KEY AND UPLOAD TO LEARNING IN ZJU"
- the key is: A(in cipher) = K(in plain)
- the plaintext is: "WANGHAOYUAN"
- the ciphertext is: "MQDWXQEOKQD"

the cracking process:

using the C++ language to design a simple program, output the 26 possible answers:

```
int main()
{
    string ciphertext;
    cin >> ciphertext;
    for (int i = 0; i < 25; i++)
    {
```

```
        for (int j = 0; ciphertext[j] != '\0'; j++)
        {
            ciphertext[j]++;
            if (ciphertext[j] > 'z' || (ciphertext[j] > 'Z' && ciphertext[j] <
'a'))
                ciphertext[j] -= 26;
        }
        cout << ciphertext << endl;
    }
}
```

after doing that, it can be clearly seen that when A = K, the ciphertext is changed into plaintext.

so the code is cracked.

ciphertext 2: by vignere algorithm

the answer:

- the ciphertext is: "ktbueluegvitnthuexmonveggmrcgxptlyhhjaogchoemqchpdnetxupbqntietiabpsmaoncnwvoutiugtagmmqsxtvxaoniiogtagmbpsmtuvvihpstpdvcrxhokvhxotawswquunewcgxptlcrxtevtubvewcnwwsxfsnptswtagakvoyyak"
- the plaintext is: "it is essential to seek out enemy agents who have come to conduct espionage(间谍活动) against you and to bribe(贿赂) them to serve you give them instructions and care for them thus doubled agents are recruited and used sun tzu the art of war(孙子兵法)"
- the key is: "cat"(that means, the first letter is c = a; second is a = a; third is t = a).
- there is nothing that I need to do according to the plaintext.

the cracking process:

1. using the Kasiski's algorithm to confirm the length of the key:

I use the length of 3 to check the times of repeating using the function below:

```
void tripletest(string cipher)
{
    string pair1, pair2;
    for (int i = 0; i < cipher.length() - 3; i++)
    {
        pair1 = cipher.substr(i, 3);
        for (int j = i + 1; j <= cipher.length() - 3; j++)
        {
```

```
            pair2 = cipher.substr(j, 3);
            if (pair1 == pair2)
            {
                cout << pair1 << ": " << j - i;
                for (int k = 2; k < j - i; k++)
                {
                    if ((j - i) % k == 0)
                        cout << " " << k;
                }
                cout << endl;
            }
        }
    }
}
```

the output is as follows:

cgx: 114 2 3 6 19 38 57

gxp: 114 2 3 6 19 38 57

xpt: 114 2 3 6 19 38 57

ptl: 114 2 3 6 19 38 57

bps: 39 3 13

psm: 39 3 13

aon: 24 2 3 4 6 8 12

cnw: 87 3 29

gta: 18 2 3 6 9

tag: 18 2 3 6 9

tag: 90 2 3 5 6 9 10 15 18 30 45

agm: 18 2 3 6 9

tag: 72 2 3 4 6 8 9 12 18 24 36

crx: 27 3 9

ewc: 18 2 3 6 9

it can be seen that, 3 appears in all the pairs, and 6 appears in most of the pairs

so we can almost confirm that, the key's length is 3.

2. using the frequency algorthm to confirm the exact key:

first, I use the code below to calculate the times of all the letters:

```
void frequency(int x, string cipher){
    int alphabet[26];
    string* subcipher = new string[x];
    for (int i = 0; i < x; i++){
        for (int j = i; j < cipher.size(); j += 3){
            subcipher[i] += cipher[j];
        }
    }
    for (int i = 0; i < x; i++){
        for (int j = 0; j < 26; j++){
            alphabet[j] = 0;
        }
        for (int j = 0; j < subcipher[i].size(); j++){
            alphabet[subcipher[i][j] - 'a']++;
        }
        for (int j = 0; j < 26; j++){
            cout << "times of " << (char)(j + 'a') << ": " << alphabet[j] << endl;
        }
    }
}
```

I list the five top letters that appears the most in each key's list:

list1:

p:8

g:7

v:7

c:6

t:5

list2:

t:10

e:9

s:7

o:6

a:4

i:4

list3:

x:10

w:5

b:5

h:5

n:5

w:5

according to the results above, we can almost confirm that in list3, x stands for e.

after that, I tried some combinations according to the frequency result, and get the correct answer:

```
void caesar(int x, string cipher, string key){//this is the test function
    string* subcipher = new string[x];
    for (int i = 0; i < x; i++){
        for (int j = i; j < cipher.size(); j += 3){
            subcipher[i] += cipher[j];
        }
    }
    for (int i = 0; i < x; i++){
        for (int j = 0; j < subcipher[i].size(); j++){
            subcipher[i][j] += 'e' - key[i];
            if (subcipher[i][j] < 'a')
                subcipher[i][j] += 26;
            if (subcipher[i][j] > 'z')
                subcipher[i][j] -= 26;
        }
    }
    for (int i = 0; i < subcipher[0].size(); i++){
        for (int j = 0; j < x; j++){
            cout << subcipher[j][i];
        }
    }
}
```

when input key "gex", I got the correct plaintext.

so if "gex" means "eee", then "cat" means "aaa", so that is the key.

ciphertext 3: using any possible algorithm

the answer:

- the ciphertext is: "MAL TIRRUEZF CR MAL RKZYIOL EX MAL OIY UAE RICF "MAL ACWLRM DYEUPLFWL CR ME DYEU MAIM UL IZL RKZZEKYFLF GH OHRMLZH""
- the plaintext is: "the password is the surname of the man who said "the highest knowledge is to know that we are surrounded by mystery""
- the key is:

cipher	plain
A	H
C	I
D	K
E	O
F	D
G	B
H	Y
I	A
K	U
L	E
M	T
O	M
P	L
R	S
T	P
U	W

cipher	plain
W	G
X	F
Y	N
Z	R

(unconcerned letter is omitted in the convey)

- the task: this password is ALBERT, who said this sentence.

the cracking process:

1. try caesar:

using this code to check if it is caesar algorithm, the answer is no.

```
void caesar(string cipher){
    for (int i = 0; i < 26; i++){
        for (int j = 0; j < cipher.size(); j++){
            if (cipher[j] >= 'A' && cipher[j] <= 'Z'){
                cipher[j]++;
                if (cipher[j] > 'Z'){
                    cipher[j] -= 26;
                }
            }
        }
        cout << cipher << endl;
    }
}
```

2. recording frequency:

using this code to record the frequency of all the letters:

```
void frequency(string cipher){
    int alphabet[26];
    for (int i = 0; i < 26; i++){
        alphabet[i] = 0;
    }
    for (int i = 0; i < cipher.size(); i++){
        alphabet[cipher[i] - 'A']++;
    }
    for (int i = 0; i < 26; i++){
        cout << "letter " << (char)(i + 'A') << ": " << alphabet[i] << endl;
```

```
    }
}
```

the results are as follows:

letter A: 7

letter B: 0

letter C: 4

letter D: 2

letter E: 7

letter F: 5

letter G: 1

letter H: 3

letter I: 6

letter J: 0

letter K: 3

letter L: 12

letter M: 9

letter N: 0

letter O: 3

letter P: 1

letter Q: 0

letter R: 9

letter S: 0

letter T: 1

letter U: 5

letter V: 0

letter W: 2

letter X: 1

letter Y: 5

letter Z: 6

3. using English grammar(and so on) to crack the code:

the detailed process is as follows(in the form of diary):

```
重新想，MAL重复很多，根据分隔符也可以明白它是一个单词，这说明加密方式是替换而非位移
那么频率分析等等都可以用，确定一对替换就可以换所有的字母
则L大概率是e，以e结尾频繁出现的单词，一般为冠词/介词，可以猜一手the
那么MAL->THE:
MAL TIRRUEZF CR MAL RKZYIOL EX MAL OIY UAE RICF "MAL ACWLRM DYEUPLFWL CR ME DYEU
MAIM UL IZL RKZZEKYFLF GH OHRMLZH"
变为:
THE _________ __ THE _______E __ THE ___ _H_ ____ "THE H__E_T _____E__E __ T_ ____
TH_T _E __E _________E_ __ ___TE__"
显然MAIM应该是that（用元音字母换一下即可），那么I->A:
THE _A______ __ THE ____A_E __ THE _A_ _H_ _A__ "THE H__E_T _____E__E __ T_ ____
THAT _E A_E _________E_ __ ___TE__"
再看ME，显然那个词应该是to（原理同上），那么E->O:
THE _A___O__ __ THE ____A_E O_ THE _A_ _HO _A__ "THE H__E_T __O__E__E __ TO __O_
THAT _E A_E ____O___E_ __ ___TE__"
26个字母顺一遍，_HO必定是who，那么可以确定U->W:
THE _A__WO__ __ THE ____A_E O_ THE _A_ WHO _A__ "THE H__E_T __OW_E__E __ TO __OW
THAT WE A_E ____O___E_ __ ___TE__"
右侧句子顺下来显然应该是that we are ..., 则Z->R:
THE _A__WOR_ __ THE __R_A_E O_ THE _A_ WHO _A__ "THE H__E_T __OW_E__E __ TO __OW
THAT WE ARE __RRO___E_ __ ___TER_"
感觉做不下去了，回头看一眼频率分布图:
E-T-A-O-I-N-H-S-R-D-L
现在还有I/N/S/D/L没找到
看R的位置，首先它有CR这个单词（排除D/L），而RICF对应的明文中显然说明R为辅音字母（排除I）
这个时候，不论R是N还是S，C都必然确定是I了（A/E/O确定，U不可能）
那么C->I:
THE _A__WOR_ I_ THE __R_A_E O_ THE _A_ WHO _AI_ "THE HI_E_T __OW_E__E I_ TO __OW
THAT WE ARE __RRO___E_ __ ___TER_"
_AI_是动词，后面跟双引号，不妨猜测其为said，R恰好对应之前猜测的范围内
那么R->S, F->D:
THE _ASSWORD IS THE S_R_A_E O_ THE _A_ WHO SAID "THE HI_EST __OW_ED_E IS TO __OW
THAT WE ARE S_RRO__DED __ __STER_"
显然第二个词是password,改掉它:
THE PASSWORD IS THE S_R_A_E O_ THE _A_ WHO SAID "THE HI_EST __OW_ED_E IS TO __OW
THAT WE ARE S_RRO__DED __ __STER_"
此时有两个想法：猜测__OW是什么，或者S_RRO__DED是什么，猜出第二个应该是surrounded:
那么K->U, Y->N:
MAL TIRRUEZF CR MAL RKZYIOL EX MAL OIY UAE RICF "MAL ACWLRM DYEUPLFWL CR ME DYEU
```

MAIM UL IZL RKZZEKYFLF GH OHRMLZH"
THE PASSWORD IS THE SURNA_E O_ THE _AN WHO SAID "THE HI_EST _NOW_ED_E IS TO _NOW
THAT WE ARE SURROUNDED __ __STER_"
加上这个线索，可以猜出_NOW是know，则D->K:
THE PASSWORD IS THE SURNA_E O_ THE _AN WHO SAID "THE HI_EST KNOW_ED_E IS TO KNOW
THAT WE ARE SURROUNDED __ __STER_"
显然引号内第三个词是knowledge,则P->L, W->G:
THE PASSWORD IS THE SURNA_E O_ THE _AN WHO SAID "THE HIGEST KNOWLEDGE IS TO KNOW
THAT WE ARE SURROUNDED __ __STER_"
卧槽，密文抄错了，highest应该是没问题的。
be surrounded by,所以GH应该是by，而EX这里对O_只剩of这个可能了。
THE PASSWORD IS THE SURNA_E OF THE _AN WHO SAID "THE HIGHEST KNOWLEDGE IS TO KNOW
THAT WE ARE SURROUNDED BY _YSTERY"
显然最后一个词是mystery了，至此所有的单词都已经对应完全了，密文为：
the password is the surname of the man who said "the highest knowledge is to know
that we are surrounded by mystery"
这句话是Albert Schweitzer说的，他的姓显然就是albert了
那么password就应当是：ALBERT