实验报告

【实验目的】

1. 通过本次实验，了解古典加密算法思想，掌握常见的古典密码。
2. 学会应用古典密码以及针对部分古典密码的破译。

【实验环境】

1. 语言：C
2. 平台：clion 2021.2 版本

【实验内容】

一、仿射密码

1. **算法流程**

   - 加密根据公式：

     $$c = (k * p + b) mod 26$$

     将每个字母代入公式运算即可得结果。

   - 解密算法是上述公式的逆运算：

     $$p = (c - b) * k^{-1} mod 26$$

2. **测试样例及结果截图：**

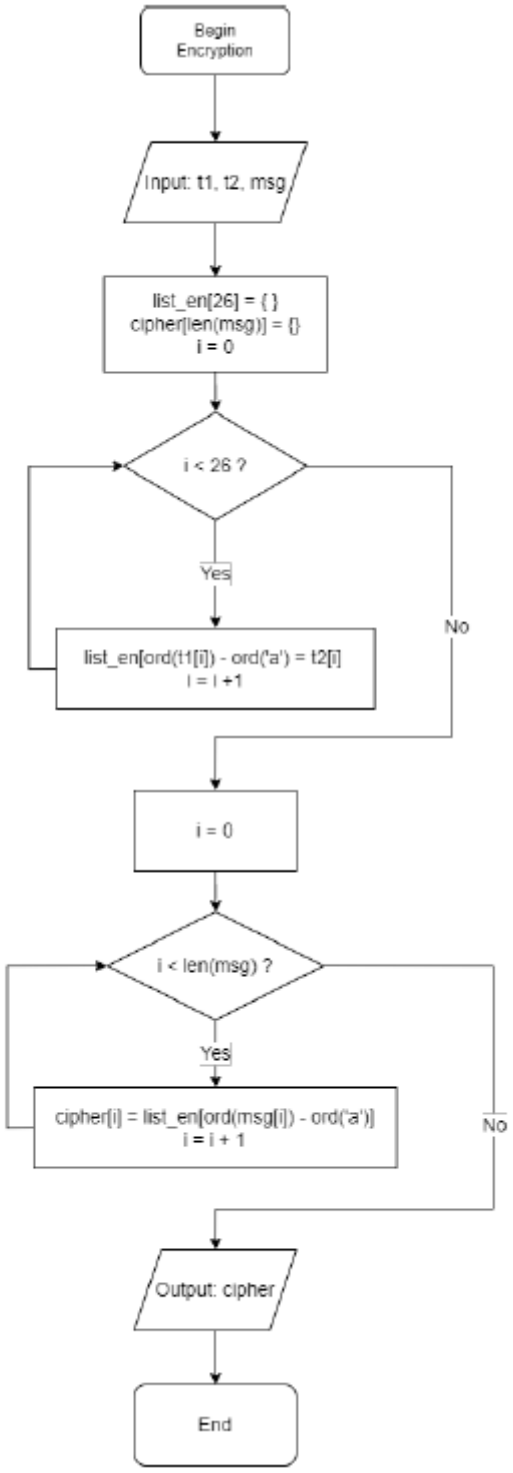
```
"E:\E_drive\clion\Project List\cryptography\Cr
9 23
abcdefg
1
xgpyhqz
Process finished with exit code 0
```

3. **讨论与思考：**

   仿射密码较为简单，编写代码过程中未遇到困难。根据其加密原理来看，仿射密码极容易破解，且无法屏蔽掉字母的统计学规律，只需要统计出频率最高的两个字母，分别对应英文字母e和t或者e和a，解方程即可破解得到密钥key(k,b)。

二、单表代替密码

1. **算法流程：**
   - 单表代替密码采用查表一一对应的方式进行加解密，在具体算法中可以采用索引与内容相对应的方式具体实现。
   - **下面给出该算法的流程图（以加密为例）：**



2. **测试样例及结果截图：**

```
"E:\E_drive\clion\Project List\cryptography\C
abcdefghijklmnopqrstuvwxyz
qazwsxedcrfvtgbyhnujmiklop
doyouwannatodance
1
wbobmkqggqjbwqgzs
Process finished with exit code 0
```

3. **讨论与思考：**

   单表代替密码**无法屏蔽掉语言的统计学规律**，所以可通过通过词频、高频词组的组合形式，来进行破解（对应选做三）。

三、维吉尼亚密码：

1. **算法流程：**

   - 设明文为P，密钥为K，密文为C。其关系为：

     $$C_i = (P_i + K_i) mod 26$$

     若明密文长度大于密钥长度，密钥可重复使用，设密钥长度为d，其关系式可以改为：

     $$C_i = (P_i + K_{i \bmod d}) mod 26$$

2. **测试样例及结果截图：**

```
"E:\E_drive\clion\Project List\cryptography\Cr
interesting
zhonghuaminzuweidafuxing
1
huhrxlmtuvthhpizhsyckovt

Process finished with exit code 0
```

3. **讨论与思考：**

   顺着思路敲代码就行了，没什么难理解的地方。

四、弗纳姆密码：

1. **算法流程：**
   - 设明文为P，密钥为K，密钥长度为d ，密文为C。其关系为：

$$C_i = P_i \oplus K_{i \bmod d}$$

2. **测试样例及结果截图：**

```
"E:\E_drive\clion\Project List\cryptography\Crypto_Experiment\
<v9nk[P
h,wX^%hy-rO?<>i.j\:y
0
TZN65~8E[K!TgnUXS2Q"

Process finished with exit code 0
```

3. **讨论与思考：**

   与维吉尼亚密码类似，较为简单，未遇到困难。

五、栅栏密码：

1. **算法流程：**
   - 设明文为P，密钥为K，密钥长度为d ，密文为C。其关系为：

$$C_i = P_i \oplus K_{i \bmod d}$$

2. **测试样例及结果截图：**

```
"E:\E_drive\clion\Project List\cryptography\Crypto
3
whateverisworthdoingisworthdoingwell
1
wtesrdnsrdneherwtogwtoglaviohiiohiwl
Process finished with exit code 0
```

```
"E:\E_drive\clion\Project List\cryptography\Crypto_Experiment\
2
hatimriprathnelhelhsoemotntawat
0
healthismoreimportantthanwealth
Process finished with exit code 0
```

```
"E:\E_drive\clion\Project List\cryptography\Crypto_Experiment
4
omaegaomaetachigaorenotsubasada
1
ogacanuamaehoobdaotirtaaemagess
Process finished with exit code 0
```

```
"E:\E_drive\clion\Project List\cryptography\Crypto_Experiment\
3
kirhauunomhiyduinaumhzntoinoemoihouenosbous
0
kiminoraihuhamouhuuzennnotomoshibinoyoudesu
Process finished with exit code 0
```

3. **讨论与思考：**

栅栏密码的难点在与解密时如何分组：

我的思路是先将明文的长度除以栅栏规定的长度，根据求得的余数进行分组：

- 求得余数不为0， 组数为商+1
- 求得余数为0， 组数为商

然后在使用两层循环找到其在明文中的位置：

```
for(group_cnt = 0; group_cnt < true_group; group_cnt++)
            for(pos = group_cnt, cnt = 0; pos < str_len;
                cnt<result.remain?(pos+=true_group):(pos+=result.ans),
cnt++)
                re[recnt++] = str[pos];
re[str_len] = '\0';
```

进而得到明文。

六、希尔密码：

1. **算法流程：**

- 设密钥为n维Hill矩阵K，将明文分组后构成$m*n$维矩阵P，密文也为$m*n$维矩阵C，三者关系为：

$$C \equiv PK \ mod \ 26$$

解密需要求出在模26意义下K的逆矩阵K^{-1}，运算关系：

$$P \equiv CK^{-1} \ mod \ 26$$

Hill密码算法设计的难点在于求K模26意义下的逆矩阵。

- 求逆矩阵的思路如下：
    1. 先构造计算矩阵行列式的函数。
        1. 先设计一个可以初始化矩阵使各位元素均为0的初始化函数

```
void init_matrix(long long int ***matrix_ptr, int r, int c);
```

2. 通过递归实现计算矩阵关于(i,j)元素的laplace_expension值与矩阵行列式的计算。
   - 利用$\mathbf{Laplace}$**展开定理**可以将行列式

$$D=\begin{vmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{13} & \cdots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \cdots & a_{nn} \end{vmatrix}$$

   按任意行任意列展开. 这里我们采用按第一列展开，得

$$D=\begin{vmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{13} & \cdots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \cdots & a_{nn} \end{vmatrix} = a_{11}A_{11}+a_{21}A_{21}+a_{31}A_{31}+\cdots+a_{n1}A_{n1}$$

   - 接着来编写 **determinant 函数**. 如果行列式为1阶，行列式的值便是 a_{11} 的值，即 matrix[0][0]; 如果行列式的阶高于1，我们采用另定义的$\mathbf{Laplace}$展开函数 laplace_expansion 来降阶，并在函数中直接展开后余子式的值. 将$\mathbf{Laplace}$展开函数定义为：

```
long long int laplace_expansion(long long int **matrix,int r,int c,int order);
```

   - 其中 r，c 分别执行展开时选定元素在行列式中的行数和列数，统一选定 $c=0$，即选取第一列元素.

     展开的结果是一个**余子式** M，用cofactor表示，由于公式中是**代数余子式** $A=(-1)^{i+j}M$，因此定义 sign = 1 用来记录该行该列代数余子式的符号，每换一行乘上 -1，于是展开式每一项的值便是

```
sign * matrix[i][0] * cofactor
```

     接下来再写determinant函数计算行列式的值：

```
long long int determinant(long long int **matrix, int order);
```

3. 这样的话，我们就能通过上面的计算得到一个矩阵行列式的值，对这个值取模便可以达到计算模26意义下的矩阵行列式的计算。
4. **接下来我们讨论如何在模意义下计算逆矩阵：**

   如果矩阵**A**有非零行列式，则这个矩阵的逆如下计算：

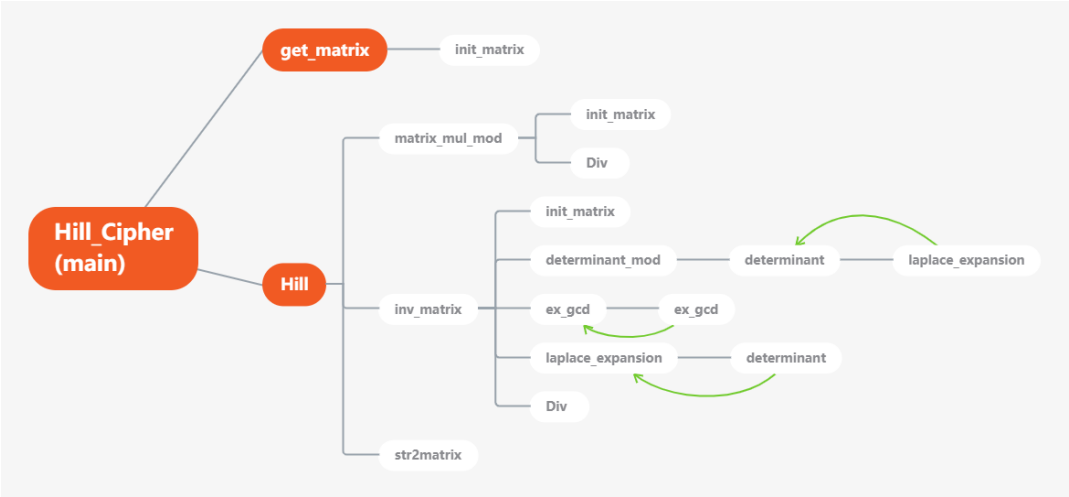
$$[A^{-1}]_{ij}=(det\ A)^{-1}(-1)^{i+j}(D_{ij})$$

   其中

   1. $(D)_{ji}$是将矩阵**A**去掉第**j**行和第**i**列后的子行列式的值

2. $det(A)$是A的行列式
3. $(det(A))^{-1}$是$det(A)\ mod\ 26$的逆

- **函数调用图如下：**



2. **测试样例及结果截图：**

```
"E:\E_drive\clion\Project List\cryptography\Crypto_Experiment\cmake-build-debug\Hill_Cipher.exe"
3
10 19 13
19 24 5
24 9 2
kibounohanatsunaidakizunagaimabokuranomunenonakaniarukarakeshitechirukotohanaiikiruchikara
1
qbqafwnsjoczodwqlusaoggjkoewyiwdzdihipdxavgyfigywxbkjdlshkgeccnmfyzmvqvensjkhduwoapnuzvlsh

Process finished with exit code 0
```

```
"E:\E_drive\clion\Project List\cryptography\Crypto_Experiment\cmake-build-debug\Hill_Cipher.exe"
3
19 13 1
9 20 20
20 17 16
qhcsoswyexlngykzjdtwpisajxt
0
miraiwokizukukibounohikarii

Process finished with exit code 0
```

```
"E:\E_drive\clion\Project List\cryptography\Crypto_Experiment\cmake-build-debu
3
5 14 20
24 5 16
12 2 15
orehaninngennwoyameruzojojo
1
gdojuxovrwaphiqewkszcxwbmjk

Process finished with exit code 0
```

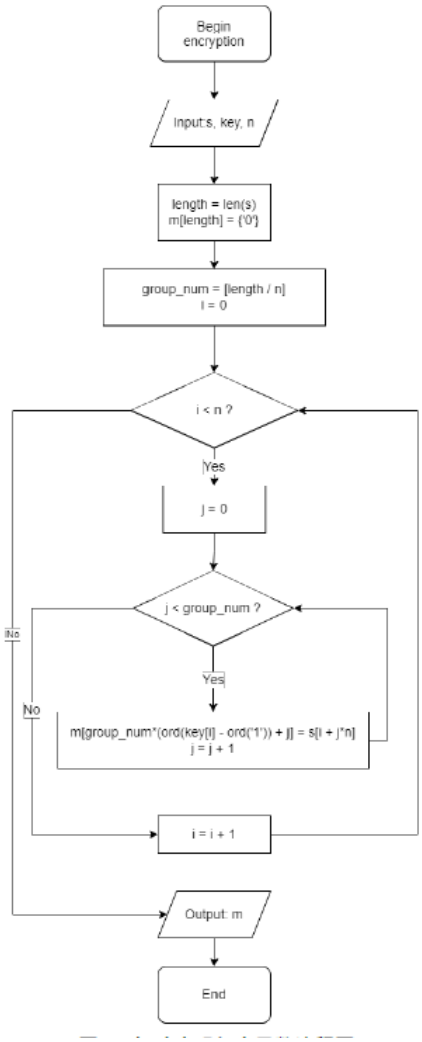
3. **讨论与思考：**

   - 加密部分没有遇到太大困难。解密部分较难的是求密钥矩阵的逆矩阵，复习过线性代数的知识后，意识到可以用求伴随矩阵的方式来求，而这有涉及到求矩阵的行列式，求行列式则利用拉普拉斯展开通过递归的方式来求解。
   - 在初次写完代码后，始终得不到结果，经过同学的提醒，发现将伴随矩阵里的元素和代数余子式的关系搞错了：伴随矩阵的(i,j)元素应该是代数余子式A_{ji}，将错误修改后成功做出本次实验。

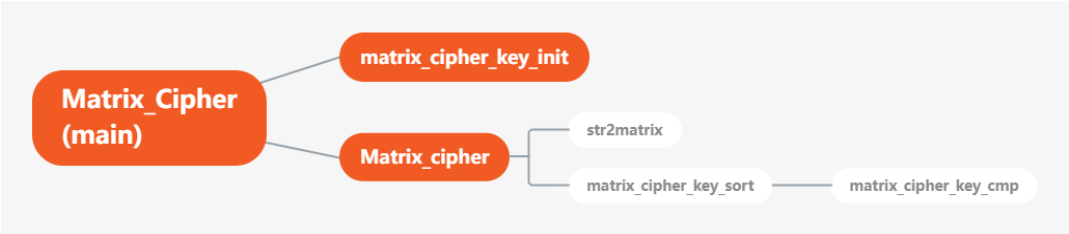
七、矩阵密码：

1. **算法流程：**

   - **流程图如下：**



   - **函数调用图如下：**



2. **测试样例及结果截图：**

```
"E:\E_drive\clion\Project List\cryptography\Crypto_Experiment\cmake-build-
7
4312567
attackpostponeduntiltwoamxyz
1
ttnaaptmtsuoaodwcoixknlypetz
Process finished with exit code 0
```

```
"E:\E_drive\clion\Project List\cryptography\Crypto_Experiment\cmake-build-debug\Matrix_Cipher.exe"
7
4312567
agbdenvaddidrghdhhnaecuianuzereeaadaiirdunidiggdauvueambianekduvwueyeuaijnmhiniliaienndvolanuilsvaenuiwnowiinanriiniixie
 xeidanitnnikighmkxamuunjuuuindeiaborenoadiwiunoaiunuggiiuiwihiaauijasnuaiaowaauiiaineghohiieedijebilnemhuidioiijeuinnaig
 zinukibudijieuinogbadibibtaadunahdahngonigdddginhtaauweyiiauedatinbodinaeduueseitdunloeoi
0
xiamashangbianmabiangundaoyourenwennineanijiukaigunaduiduiduiaduiduiudixiwangniduiniderenshengyeshizhegetaiduhaobuhaodao
shihourenjiawenanizenmemeiyoucheniaduiduiduinikaibaijiuwanlebeiduibuduizaidanweilitourenjiaweneinizenmebuganhuoniaduidui
duiniqiaodongziligaixiaobeierniaduiduiduiduijiuwanlebeinisuishiduibeitiantiangenatangnaaduiduiduikaigunjiuwanlagundaoshu
oshenmewodoukaibai
Process finished with exit code 0
```

3. **讨论与思考：**

矩阵密码和栅栏密码比较类似，都是通过分组、建立明密文的行列关系来实现的。

八、加法密码の字母频率攻击(选做一)

1. **算法流程：**

   - 加法密码的加密原理为：$c = (m + k) mod\ 26$。
     由于在英文中字母e的出现频率最高，所以只需要统计密文中各字母出现的次数，其中出现频率最高的字母即为e的密文，再根据公式计算：$k = (c - m) mod\ 26$（m为字母e对应的数字），即可得出密钥k。

2. **测试样例及结果截图：**

```
"E:\E_drive\clion\Project List\cryptography\Crypto_Experiment\cmake-build-debug\Attack_Additive_Cipher.exe"
chnqyhnsmypyhnsmypyhnbyspinyxgswcnsnbyqilmnjfuwynifcpychugylcwuguchcmmoymmesbcablunyizpcifyhwyuhxgilyjyijfuf
 yfmywuhnxyhscnmnloyvonypylsvixsmncffquhnmnifcpypylynbcmwcnsufqusmainujligcmyzilsiogcabnvyufcyuhcffomcihvonc
 aichacnmwcnsxlyugmuhxcguvcaxlyugyl
20
Process finished with exit code 0
```

3. **讨论与思考：**

由此可见，单纯的加法密码（即凯撒密码）极易破解。

九、希尔密码の已知明文攻击(选做二)

1. **算法流程：**

   - 设希尔密钥矩阵K维数为n，则可将n组明文及其对应的密文按行排列，组成n维方阵M和C，则运算关系为：

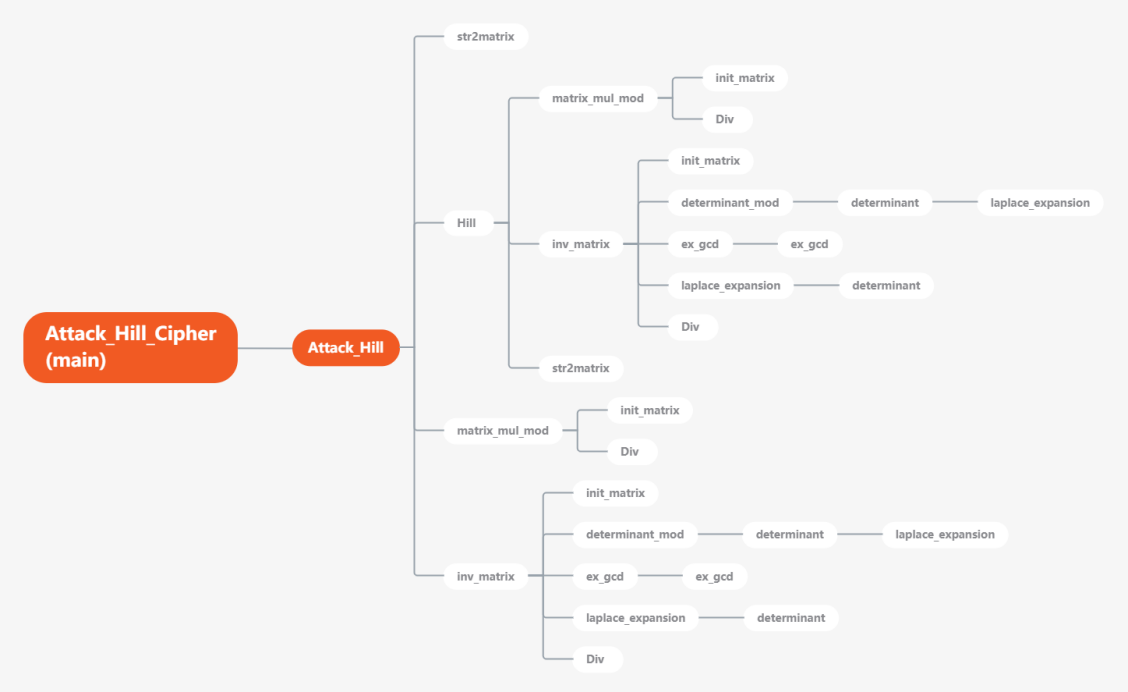
   $$C = MK\ mod\ 26$$

   - 若明文矩阵M可逆，则可通过运算得到密钥矩阵：

$$K = M^{-1}C \ mod \ 26$$

- 所以该题关键在于，如何得到可逆的明文矩阵M，可以通过逐一尝试组合分组后的明文得到矩阵M，再求det (M)，若该行列式的值不为0,2,13，则该矩阵可用于攻击。

2. **函数调用图：**



3. **测试样例及结果截图：**

```
"E:\E_drive\clion\Project List\cryptography\Crypto_Experiment\cmake
3
thisistestfortheknownplaintextattack
wvlomkzsbyjjwdisydygrqtfrpnfxnhthysu
7 2 15
25 25 6
0 2 19

Process finished with exit code 0
```

```
"E:\E_drive\clion\Project List\cryptography\Crypto_Experiment\cmake
3
horakanntandarodaremokizutsukanaisekainokanseida
uodwighujxxlnkacqnkyckdvreokussfmoekcjxygeunwhce
2 11 24
23 6 24
12 25 21

Process finished with exit code 0
```

```
"E:\E_drive\clion\Project List\cryptography\Crypto_Experiment\cmake-build-debug\Atta
3
itsukawatashitachikanarazukokowamanninnnishitemisemasu
bxteogdozzadfnxzlrnixzhxakpygugacanaajiuydbqeowmswimaq
0 6 1
3 5 9
7 2 7

Process finished with exit code 0
```

4. **讨论与思考：**
    - 思考题：m维Hill密码的破解效率。假定已得到可逆的明文矩阵M，矩阵求逆的时间复杂度为$O(m^3)$，矩阵乘法的时间复杂度为$O(m^3)$，所以整体的时间复杂度为：$O(m^3)$+$O(m^3)$=$O(m^3)$。
    - 此外，经过助教提醒，发现在求行列式函数中，对于3维矩阵直接输出结果，对于程序的运行速度有较大提升。原因是，在寻找可逆的明文矩阵M、以及求逆矩阵的过程中，会调用较多的求行列式的函数，而行列式是通过递归的方式来求，递归调用会导致递归到2维矩阵的情形变得非常多，所以如果单独判断2维矩阵的情形能大幅提高算法效率。

【收获与建议】

思考题：古典密码体制的缺陷与不足。

- 古典密码体制，包括多表代替密码在内，都无法完全消除语言的统计特性，因此现在均已破解，已不再使用。
- 本次实验让我对于古典密码体制有了更深刻的认识，通过实现Hill密码的加解密，我复习巩固了线性代数的相关知识，通过编写部分密码体制的攻击代码，了解了一些破译密码的攻击手段。同时，也感受到了上次实验与本次实验的相互联系：在本次实验中，也用到了不少数论的相关知识和上周写过的函数。希望下次实验依然能有所收获。