Programming：

Time Complexity：分析所给代码的时间复杂度，一重循环。

Square Sum：把 25000 拆分成两个数的平方和，第一个数从 0 到 160 遍历，计算第二个数，如果满足要求则输出。

```
int main(){
    int x = 25000;
    int a,b;
    for(a = 0; a <= 160; a++){
        int y = x - a * a;
        b = sqrt(y);
        if(a * a + b * b == x){
            printf("%d,%d\n",a,b);
        }
    }
    return 0;
}
```

Something In Common：辗转相除法求两个数的最大公因数。

```
def ex_gcd(a, b, arr):
    if b == 0:
        arr[0] = 1
        arr[1] = 0
        return a
    g = ex_gcd(b, a % b, arr)
    t = arr[0]
    arr[0] = arr[1]
    arr[1] = t - a // b * arr[1]
    return g


if __name__ == "__main__":
    x = ex_gcd(21525625, 30135875, [0, 1, ])
    print(x)
```

Loop In A Loop：加密过程是依次将第 i 位字符移到字符串末尾，i 从 1 到字符串长度。解密过程则将字符串末尾的字符移到第 i 位，i 从字符串长度到 1。

```
int main(){
    char str[] = "CFb5cp0rm1gK{1r4nT_m4}6";
    char tmp;
    for(int i = strlen(str) - 1; i >= 0; i--){
        for(int j = strlen(str) - 1; j > i;j--){
            tmp = str[j];
            str[j] = str[j - 1];
```

```
                str[j - 1] = tmp;
            }
        }
        printf("%s",str);
        return 0;
    }
```

Cryptography：

Passwd： knight:x:708697c63f7eb369319c6523380bdf7a:/home/junior:/bin/zsh 由 Linux 用户的格式可知，密码加密结果为 708697c63f7eb369319c6523380bdf7a，共 128 位。考虑 md5 算法加密，则解密可得结果为 exploit。

404 Not Found：将所给网址后半段的密文 03MTJ3M2NjcfBDdfR2Mz42X1BTefN3MNFDdz0EMTtnRUN0S 颠倒后进行 base64 解码得到 flag：KCTF{S0M3t1M3s_y0u_n33d_t0_r3v3rS3}。

Jumble：根据加密算法，参考 Loop In A Loop 的算法对密文进行解密得到明文 S0NURnt5MHVfZzB0X20zfQ==，末尾有两个“=”补位，考虑用 base64 解码得到 flag：KCTF{y0u_g0t_m3}。

The Pairs：采用 Twin-Hex Cypher 进行加密，每两个字符为一组根据字母表进行替换得到密文（https://www.calcresult.com/misc/cyphers/twin-hex.html）。解密得到 flag：KCTF{Th1s_1s_Tw1n_H3x_Cypher} 。

RSA-One：对于私钥中未知的那一位，遍历 ascii 码中的所有大小写英文字母，使用该密钥对密文进行解密，直到解密结果为 KCTF 开头的字符串，此时输出该结果。

```
from Crypto.Util.number import long_to_bytes
from Crypto.PublicKey import RSA
import string
import re
a='''-----BEGIN RSA PRIVATE KEY-----
MIIEowIBAAKCAQEAyiytHt1AKzYLwZPm1dd9uT7LgsqVj0eSLpheNd0H4xyiZCYG
ZtRYnNtGNnq7A/ubyFalExm61QNewfy71h6xhM/'''

b = '''IEIoNT0VfMOIzcq0Jmoh+v6k
6/x/3GRkk/vLVolsLbkOKd4aorPMwEsZX4vMd+Sga5Mz0tx5xLFZsbl0r1vvtBl7
CtC/ojWX4+/RSGuaUVVayrU32kyCjJo3hniSaY2EvSXXHdE6nOKkF725LVrnOOIz
1/n9CYrYPV6ESEBdwS7VOen8uPwh5cFGHOV49ofmvVZNvcV2qoKFjY5UXf8fDzZ+
jBzWiCukE+3WFwgEYaBGg/a6HomkobpDqxkrYwIDAQABAoIBABht45FaLLnL8wm2
BGuMeV2b791i+0Vv4YMN2Dxr89sGh7zQN2/PctGpUUed9uEZUw6XIaU4M7IvkRCh
qFTMKqkgrVd4hwE/20vTGMG9H52Qr4Bzqpv1S8Hmw5x6DWzseAziUorOkqtcTH5j
1LIN42wNTTESfW2aRIB26Z6nCSlzHD8jpBYlrBFNsXydApEtA86PPtgs8MUsABFa
Rhy6VG9rNfzaBeRDX1m1lX+yNkqPb3xgABeYgURYgUneiTY/S5GrFfrtRAnLWVm4
```

```
audCUkxvF8OV0vJnazcMUopleBonMH2FCl3vKAjTX2xq9X24PeNXDg6SfiEEuI/g
EDtJO1ECgYEAzwBWVwbx/lvc5PP3oYXRr9IpflZ3Z9xSyopY0KpOakAXn6717x6i
s/1DwGvpmFBqUd25vhcn9ztj18GtMCtZ4dNvvyGpPwvM41Z1RVHY5REfC7sgBp8W
0N+IVR2QlyU3pjoS5t19O3g48fhOp8o3wsZ+05RpLtUhNXe0yHxk7fsCgYEA+gfZ
aCr+dgzHfdBOEwwozaRpJANchnGeILSgZZEeYmyE0RuBcatpwxKs+jG82mWYnosN
KR5CZZiPn/laySUQEB5H6Cg/OQDVyj5r49adc2H8hTCluaXtiVyxA3JqV8Ixc9TM
cRWJZdokaDbkyNXCuUuTMinzWjrNBKBZ+zg5w7kCgYBQkjwJEb39mHoJb+CSMUkl
23KlJzjA52QeS+04AyIUfy/yyqIVWeJQlqLZcedxjtNjXB9hGxhGRgqdv1gO6MDK
gob7aTm8PXaZglyRB8OZnals4oAbs66ozGj/YEuYWTco72/OBqYpEKlxnYnYC4Da
wnI5Hoo2XWTYr+hhJPIQIwKBgAxMxo0xUENObaHq1WxqdLdpFyMGZ07V2AmT2TAl
63C8FeyThdKptBI8oPXN7JRx2wgxnvwe2PVWg/pCsgyjHh8s3iy1jianu9yvJW+X
5zb94wZKVlzDpOPVA4A/6KtYikZAea42eQPhr1jRGoAmw+WJqjwVhDs0GVHY8ZRC
N9VBAoGBAJTZwrY+tZkNzURk9JLWzrevfD6BpYrQ0jchaGtzdgjdOpHo3++cdUag
9oQ8ZNKaUVDm3lyzUhO41Hw7xMmmW8JwsVvKdrRL+ZG12Ts/uiy1P0DY+HsNMr9d
xqG9YAHVmm4iJzcHeMdzLwmzR6D/x6+k2cFWwox6PxvA7ikJQEYr
-----END RSA PRIVATE KEY-----
'''
alphabet_string = string.ascii_lowercase
alphabet_string2 = string.ascii_uppercase
ct =
372786001527198484132415976592853742211159156316870324627453405821286014832498
492653312983497201550123716433759628352858631460813788307599175721042441522806
037093035940687232794557723019054139542415842112954053279957671684443707769546
844129836128128728150073110751011091979231639459880263174335603746192568262938
664048369802709681385743418245456079376014547420215771733836091813096278054842
453733230521728843402743697415126026847116960362828022174202189008831185630074
813243405777657730898071840708390579271851769220628694083772113283373944540041
156501363873022159914259462126899071149037404338750197102426088408809 6
alphabets = alphabet_string+alphabet_string2
length = len(alphabets)
for i in range(0,length):
    key = RSA.importKey(a+alphabets[i]+b)
    n = key.n
    d = key.d
    m = pow(ct,d,n)
    m = long_to_bytes(m)
    if (m.find('KCTF') != -1):
        flag = re.compile('KCTF+.*')
        print(flag.findall(m)[0])
```

AlphabetknockCode：参见 tap code 的加密方式（https://www.boxentriq.com/code-breaking/tap-code），对 24 位的密码表构造为如图所示的 4×6，解出明文为 SECRETKNIGHT。

	1	2	3	4	5	6
1	A	B	C/K	D	E	F
2	G	H	I	J	L	M
3	N	O	P	Q	R	S
4	T	U	V	W	X	Y/Z

Tony Stark Needs Help：根据加密算法逆向求解，利用如下算法对两段密文解密得到 flag：KCTF{AREA51_TonyTheBadBoyGotScaredOfTheFatBoy}。

```
# cipher = "IihsIb_7[^7is<inH][l_^D`Ib_;[n7iu"
cipher = "6G:653"
lst = ['T3NR1NG$', 'T3nR1ng$', 'TenRings', 'T3nR!ngs', 'T3nR!ng$', '73NR1GN$', '73nRing$', 'T3nR!nG$']
secarr = []
keyarr = []
x = 0

def keyfunc(key,keyarr,x):
    for character in key:
        keyarr.append(ord(character))

    for i in keyarr:
        x += i

for key in lst:
    keyarr = []
    for character in key:
        keyarr.append(ord(character))
    x = 0
    for i in keyarr:
        x += i

    ciparr = []
    for character in cipher:
        ciparr.append(ord(character))

    if x - 1 % 2 == 0:
        ciparr[-1] = ciparr[-1] - 3
    else:
        ciparr[-1] = ciparr[-1] - 2

    for i in range(len(ciparr)):
        if 91 <= ciparr[i] <= 116:
            ciparr[i] = ciparr[i]+6
        else:
```

```
            if 54 <= ciparr[i] <= 79:
                ciparr[i] = ciparr[i]+11

    for val in ciparr:
        print(chr(val),end='')
    print()
```

Festival：根据加密算法，将密文从中间分为前后两段，按照加密思路逆向推回去即可得到 flag：KCTF{feistel_cipher_ftw}。

```
m, n = 21, 22

def f(word, key):
    out = ""
    for i in range(len(word)):
        out += chr(ord(word[i]) ^ key)
    return out


ct = open("cipher.txt", "r").read()
L, R = ct[0:len(ct) // 2], ct[len(ct) // 2:]
x = "".join(chr(ord(f(R, n)[i]) ^ ord(L[i])) for i in range(len(L)))
y = f(R, 0)

L, R = y, x
x = "".join(chr(ord(f(R, m)[i]) ^ ord(L[i])) for i in range(len(L)))
y = f(R, 0)

flag = x + y
print(flag)
```

Tony Stark Needs Help Again：根据加密算法逆向求解，利用如下算法对密文解密得到 flag：KCTF{MwUUhKU@Uk-F@pmAn-will-K1lL-All-TH3-Av3nGeR$}。

```
import base64
cipher = "JUglWEMyZlo9MkpCPSgoWj1pKDJaPSgoe1M9Q1oiayNYPV1Kl0paLUpKU0M="

l_c = ''.join(chr(c) for c in range (97,123))
u_c = ''.join(chr(c) for c in range (65,91))
l_e = l_c[13:] + l_c[:13]
u_e = u_c[13:] + u_c[:13]
so = ""
cipher = base64.b64decode(cipher).decode("ascii")
def get_key(dic, val):
    for key, value in dic.items():
        if val == value:
```

```
                return key
        return "key doesn't exist"

d2    = { "~" : "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",
          "?" : "a",
          "/" : "b",
          "5" : "c",
          "1" : "d",
          "4" : "e",
          "2" : "f",
          "3" : "g",
          "9" : "h",
          "0" : "i",
          "8" : "j",
          "6" : "k",
          "7" : "l",
          "K" : "m",
          "Q" : "n",
          "F" : "o",
```

```
"V" : "p",
"X" : "q",
"D" : "r",
"S" : "s",
"A" : "t",
"J" : "u",
"N" : "v",
"M" : "w",
"P" : "x",
"I" : "y",
"T" : "z",
"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" : ">",
"a" : "?",
"b" : "/",
"c" : "5",
"d" : "7",
"e" : "3",
"f" : "2",
"g" : "4",
```

```
            "h" : "9",
            "i" : "0",
            "j" : "8",
            "k" : "6",
            "l" : "1",
            }
d1    = { "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" : ">",
            "a" : "?",
            "b" : "/",
            "c" : "5",
            "d" : "1",
            "e" : "3",
            "f" : "2",
            "g" : "4",
            "h" : "9",
            "i" : "0",
            "j" : "8",
            "k" : "6",
            "l" : "7",
```

"m" : "K",
"n" : "Q",
"o" : "F",
"p" : "V",
"q" : "X",
"r" : "D",
"s" : "S",
"t" : "A",
"u" : "J",
"v" : "N",
"w" : "M",
"x" : "P",
"y" : "I",
"z" : "T",
"~" : "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",
"?" : "a",
"/" : "b",
"5" : "c",
"1" : "d",

```
            "3" : "e",
            "2" : "f",
            "4" : "g",
            "9" : "h",
            "0" : "i",
            "8" : "j",
            "6" : "k",
            "7" : "l",
            }
vas1 = ""
saa2t = ''
for i in cipher:
    saa2t = get_key(d2,i) + saa2t
arr2= [ord(c) for c in saa2t]
print(arr2)
for c in arr2:
    tmp = get_key(d1,chr(c - 1))
    if tmp in l_c:
        print(l_e[l_c.find(tmp)],end='')
    elif tmp in u_c:
        print(u_e[u_c.find(tmp)],end='')
    else:
        print(tmp,end='')
print()
for c in arr2:
    tmp = get_key(d1,chr(c - 2))
    if tmp in l_c:
        print(l_e[l_c.find(tmp)],end='')
    elif tmp in u_c:
        print(u_e[u_c.find(tmp)],end='')
    else:
        print(tmp,end='')
```