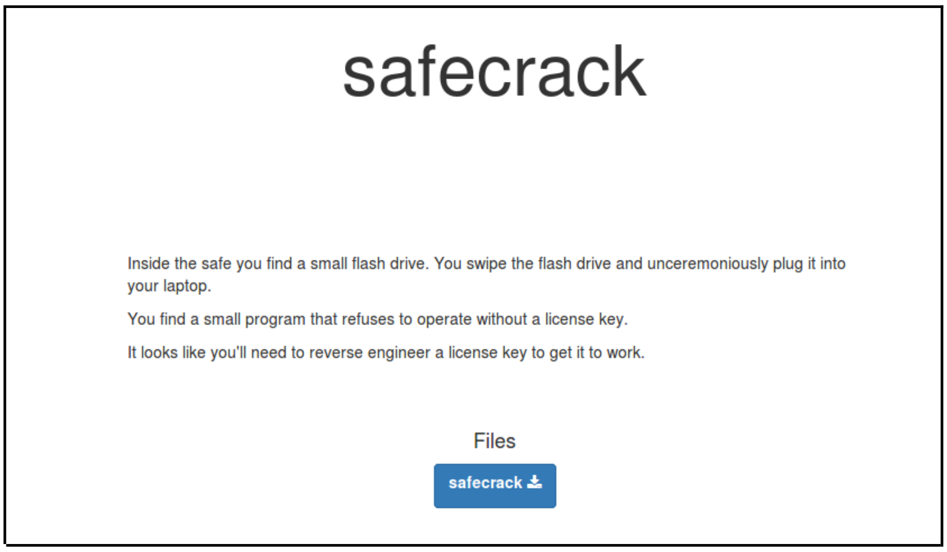
REVERSING

Author: **mucomplex**

Email:mucomplex@gmail.com

Team: **3l173Gh057**

Challenge:



safecrack

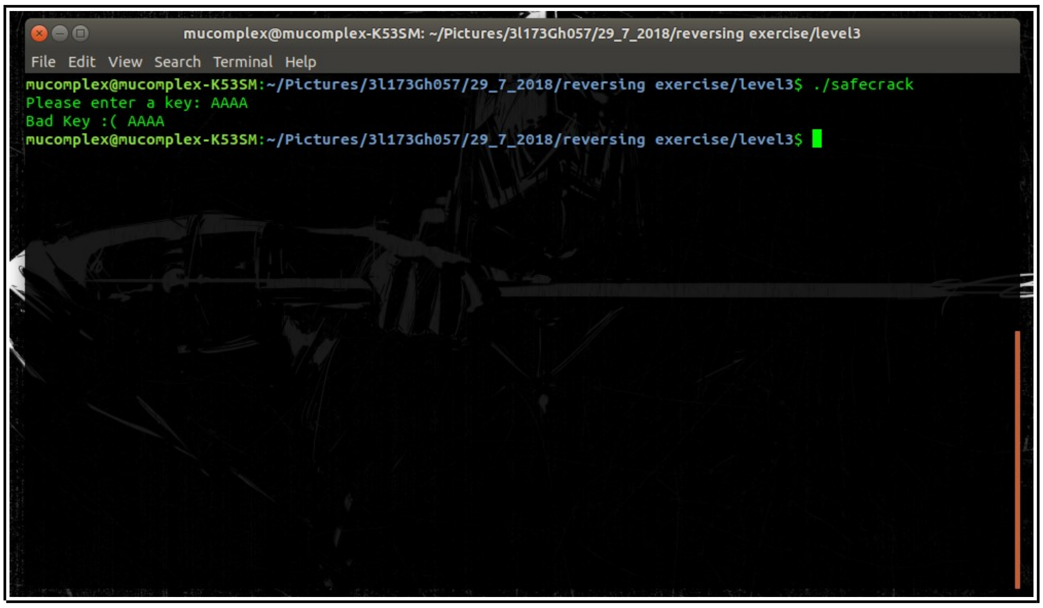
Inside the safe you find a small flash drive. You swipe the flash drive and unceremoniously plug it into your laptop.

You find a small program that refuses to operate without a license key.

It looks like you'll need to reverse engineer a license key to get it to work.

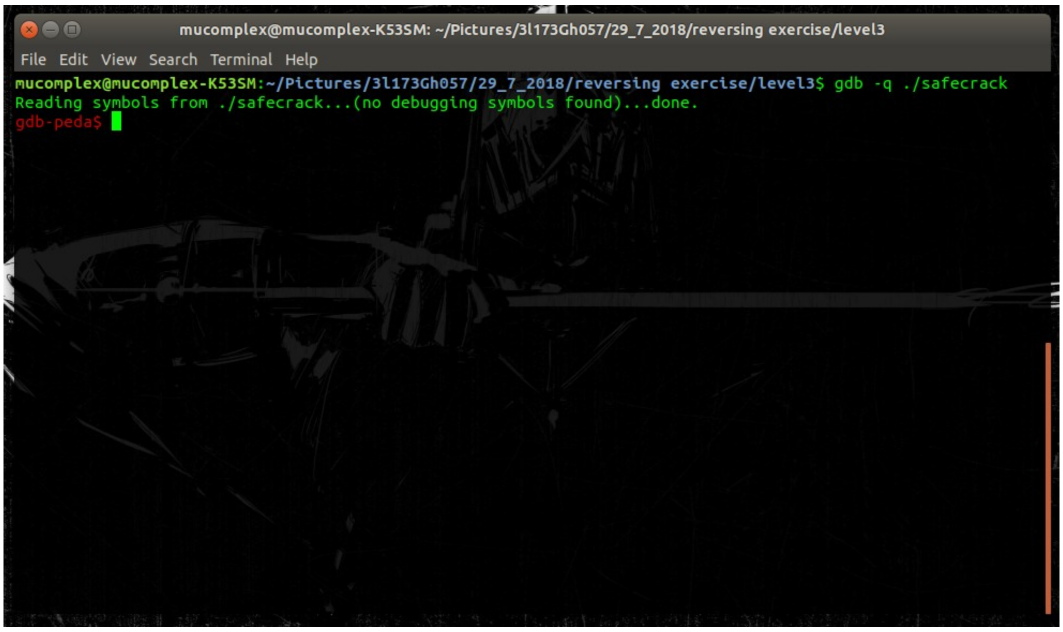
Files

Test the program to see how it works.



```
mucomplex@mucomplex-K53SM:~/Pictures/3l173Gh057/29_7_2018/reversing exercise/level3$ ./safecrack
Please enter a key: AAAA
Bad Key :( AAAA
mucomplex@mucomplex-K53SM:~/Pictures/3l173Gh057/29_7_2018/reversing exercise/level3$
```

Run the program with gdb-peda.

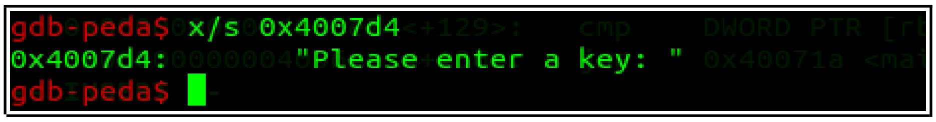

mucomplex@mucomplex-K53SM: ~/Pictures/3l173Gh057/29_7_2018/reversing exercise/level3
File Edit View Search Terminal Help

```
mucomplex@mucomplex-K53SM:~/Pictures/3l173Gh057/29_7_2018/reversing exercise/level3$ gdb -q ./safecrack
Reading symbols from ./safecrack...(no debugging symbols found)...done.
gdb-peda$
```

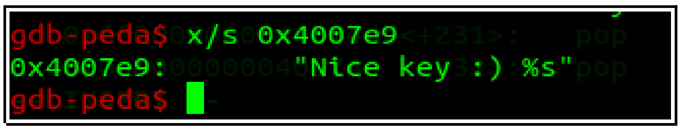
After run the program, I try to study the code.

```
gdb-peda$ disassemble main
Dump of assembler code for function main:
   0x0000000000400666 <+0>:     push   rbp
   0x0000000000400667 <+1>:     mov    rbp,rsp
   0x000000000040066a <+4>:     push   rbx
   0x000000000040066b <+5>:     sub    rsp,0x48
   0x000000000040066f <+9>:     mov    DWORD PTR [rbp-0x44],edi
   0x0000000000400672 <+12>:    mov    QWORD PTR [rbp-0x50],rsi
   0x0000000000400676 <+16>:    mov    rax,QWORD PTR fs:0x28
   0x000000000040067f <+25>:    mov    QWORD PTR [rbp-0x18],rax
   0x0000000000400683 <+29>:    xor    eax,eax
   0x0000000000400685 <+31>:    mov    edi,0x4007d4          # "Please enter a key: " is store in this address
   0x000000000040068a <+36>:    mov    eax,0x0
   0x000000000040068f <+41>:    call   0x400530 <printf@plt>    #printf the edi (destination index)
   0x0000000000400694 <+46>:    mov    rdx,QWORD PTR [rip+0x2009b5]     # 0x601050 <stdin@@GLIBC_2.2.5>
   0x000000000040069b <+53>:    lea    rax,[rbp-0x30]
   0x000000000040069f <+57>:    mov    esi,0xd
   0x00000000004006a4 <+62>:    mov    rdi,rax
   0x00000000004006a7 <+65>:    call   0x400550 <fgets@plt>     # fgets ( get input from user)
   0x00000000004006ac <+70>:    mov    DWORD PTR [rbp-0x38],0x0
   0x00000000004006b3 <+77>:    mov    DWORD PTR [rbp-0x34],0x0
   0x00000000004006ba <+84>:    jmp    0x4006d0 <main+106>  # jmp ( must jump to address 0x00000000004006d0 which main+106 )
   0x00000000004006bc <+86>:    mov    eax,DWORD PTR [rbp-0x34]
   0x00000000004006bf <+89>:    cdqe
```

```
0x00000000004006c1 <+91>:       movzx  eax,BYTE PTR [rbp+rax*1-0x30]
0x00000000004006c6 <+96>:       movsx  eax,al
0x00000000004006c9 <+99>:       add    DWORD PTR [rbp-0x38],eax
0x00000000004006cc <+102>:      add    DWORD PTR [rbp-0x34],0x1
0x00000000004006d0 <+106>:      mov    eax,DWORD PTR [rbp-0x34]
0x00000000004006d3 <+109>:      movsxd rbx,eax
0x00000000004006d6 <+112>:      lea    rax,[rbp-0x30]
0x00000000004006da <+116>:      mov    rdi,rax
0x00000000004006dd <+119>:      call   0x400510 <strlen@plt>       # seem like its compare length strings right?. see the below code, cmp
0x00000000004006e2 <+124>:      cmp    rbx,rax                     # compare rbx,rax ... lets guess our input lenght will be in rax
0x00000000004006e5 <+127>:      jb     0x4006bc <main+86> # jb mean (jump below) . so if the rbx and rax is same lenght? we not jump
0x00000000004006e7 <+129>:      cmp    DWORD PTR [rbp-0x38],0x539          # compare rbp-0x38 and 0x539(1337 in decimal) ?
0x00000000004006ee <+136>:      jne    0x40071a <main+180>                 # if equal it not jump.
0x00000000004006f0 <+138>:      lea    rax,[rbp-0x30]
0x00000000004006f4 <+142>:      mov    rdi,rax
0x00000000004006f7 <+145>:      call   0x400510 <strlen@plt>               # seem like it compare length again
0x00000000004006fc <+150>:      cmp    rax,0xc                                     # compare rax with 0xc (12 in decimal)
0x0000000000400700 <+154>:      jne    0x40071a <main+180>
0x0000000000400702 <+156>:      lea    rax,[rbp-0x30]
0x0000000000400706 <+160>:      mov    rsi,rax
0x0000000000400709 <+163>:      mov    edi,0x4007e9                        # Nice key :) %s
0x000000000040070e <+168>:      mov    eax,0x0
0x0000000000400713 <+173>:      call   0x400530 <printf@plt>
0x0000000000400718 <+178>:      jmp    0x400730 <main+202>
0x000000000040071a <+180>:      lea    rax,[rbp-0x30]
0x000000000040071e <+184>:      mov    rsi,rax
0x0000000000400721 <+187>:      mov    edi,0x4007f8
0x0000000000400726 <+192>:      mov    eax,0x0
0x000000000040072b <+197>:      call   0x400530 <printf@plt>
0x0000000000400730 <+202>:      mov    eax,0x0
0x0000000000400735 <+207>:      mov    rcx,QWORD PTR [rbp-0x18]
0x0000000000400739 <+211>:      xor    rcx,QWORD PTR fs:0x28
0x0000000000400742 <+220>:      je     0x400749 <main+227>
0x0000000000400744 <+222>:      call   0x400520 <__stack_chk_fail@plt>
0x0000000000400749 <+227>:      add    rsp,0x48
0x000000000040074d <+231>:      pop    rbx
0x000000000040074e <+232>:      pop    rbp
0x000000000040074f <+233>:      ret
```



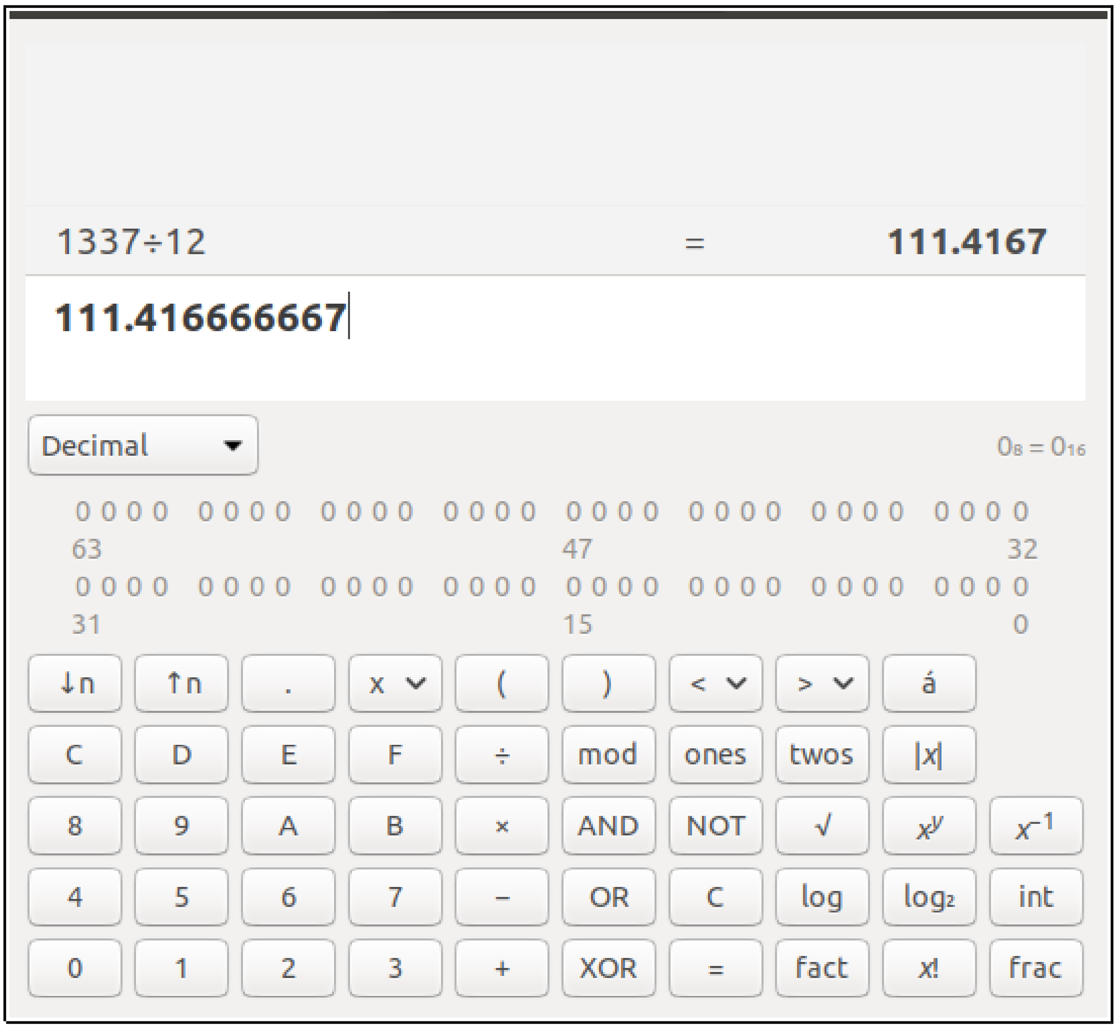
```
gdb-peda$ x/s 0x4007d4
0x4007d4:	"Please enter a key: "
gdb-peda$ 
```



```
gdb-peda$ x/s 0x4007e9
0x4007e9:	"Nice key :) %s"
gdb-peda$ 
```

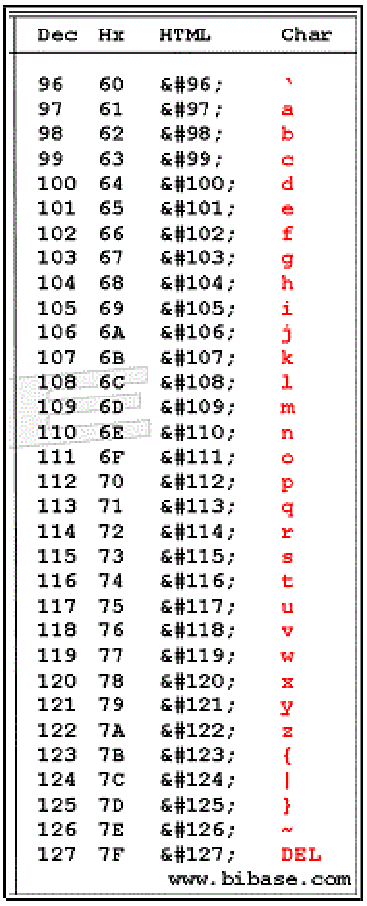
by looking at the code, we can conclude the input take 12 char length and value must be 1337 in decimal

by divide 1337 with 12. we get nearly 111.41



```
1337÷12                =                111.4167
111.416666667
```

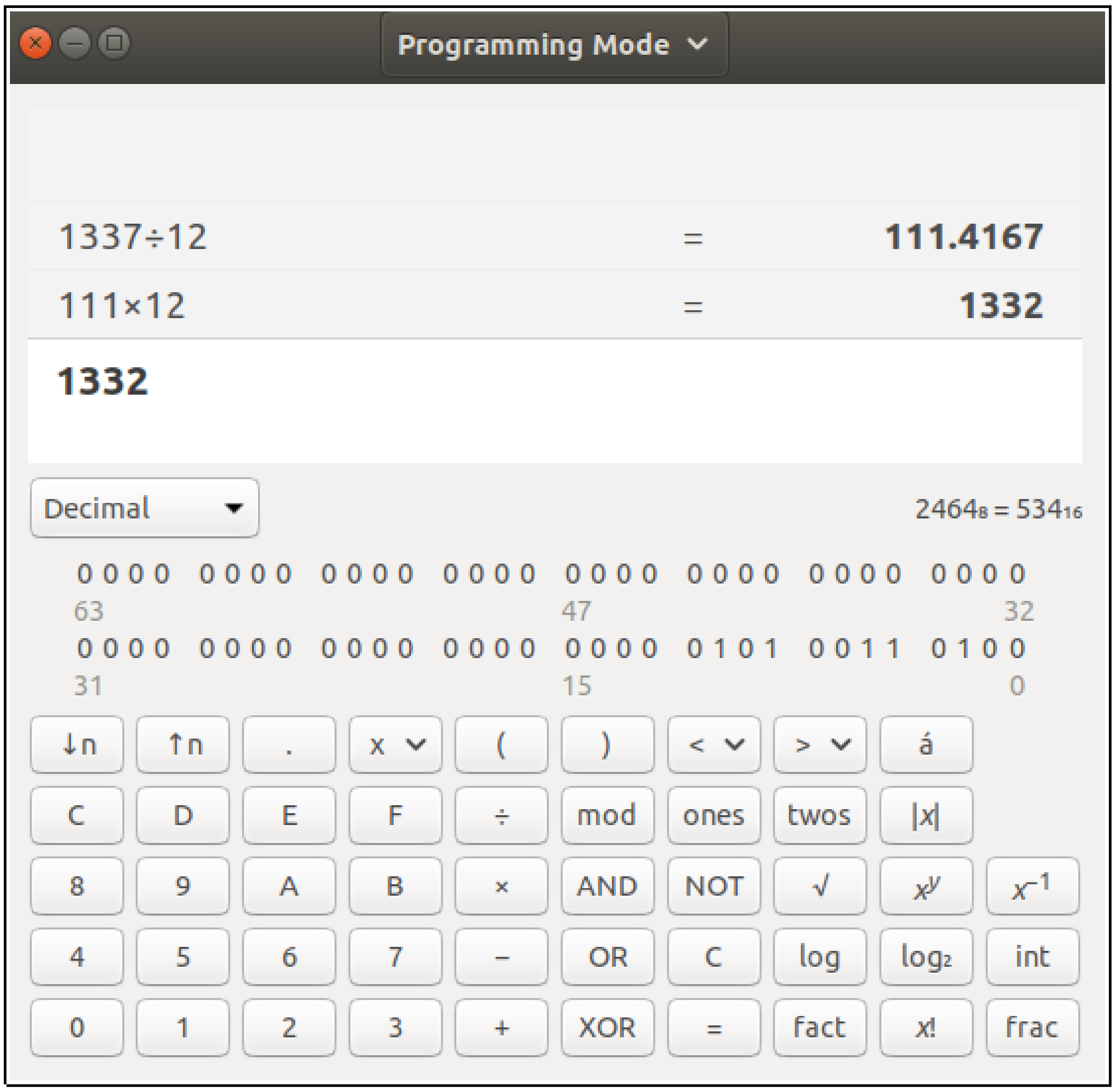
then I try to look in ascii table.



Dec	Hx	HTML	Char
96	60	&#96;	`
97	61	&#97;	a
98	62	&#98;	b
99	63	&#99;	c
100	64	&#100;	d
101	65	&#101;	e
102	66	&#102;	f
103	67	&#103;	g
104	68	&#104;	h
105	69	&#105;	i
106	6A	&#106;	j
107	6B	&#107;	k
108	6C	&#108;	l
109	6D	&#109;	m
110	6E	&#110;	n
111	6F	&#111;	o
112	70	&#112;	p
113	71	&#113;	q
114	72	&#114;	r
115	73	&#115;	s
116	74	&#116;	t
117	75	&#117;	u
118	76	&#118;	v
119	77	&#119;	w
120	78	&#120;	x
121	79	&#121;	y
122	7A	&#122;	z
123	7B	&#123;	{
124	7C	|	|
125	7D	&#125;	}
126	7E	&#126;	~
127	7F	&#127;	DEL

www.bibase.com

The ascii ‘o’ is equal to 111. then use the calculator to calculate value ‘o’ times 12.



```
1337÷12 = 111.4167
111×12 = 1332
1332
```

that mean we less 5 value from 1337. by adding 5 to last value of ‘o’ , it will become ‘t’.

so the total ‘o’ x 11 + ‘t’ will be 1337

After testing the program, the logic will be like this is python:

```
key = input(“Please enter a key:”)
if (key == *****):
        print(“ Nice key :) “)
else:
        print(“Bad Key :( “)
```

This is our first guess.

After looking into GDB , the logic will be like this:

```
key = input(“Please enter a key:”)
result = 0
for  number in range(len(key)):
        result += ord(key[number])
key = result
if (key == 1337):
        print(“ Nice key :) “)
else:
        print(“Bad Key :( “)
```

Solution:

I create small keygen:

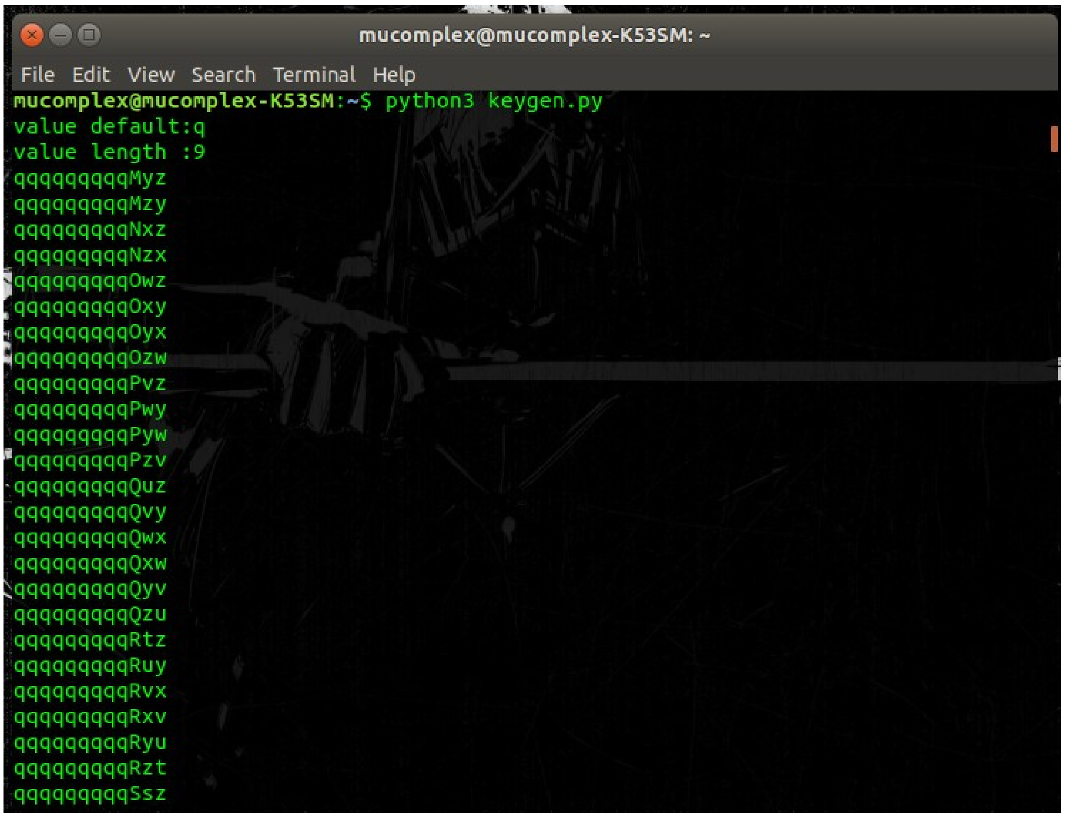
```
import string
import itertools
import os
'''
usage:

value default:q
value length :9

shorter lenght value, longer the result will be.
'''

#input
default = input("value default:")
length  = input("value length :")
#default lenght is 12 - length default will result mutate value
mutate = 12 - int(length)
#get list upper and lowercase alphabet
list_alphabet = string.ascii_uppercase + string.ascii_lowercase

items = itertools.permutations(list_alphabet,mutate)
for item_list in items:
    result = ord(default)*int(length)
    for value in item_list:
        result += ord(value)
    if result == 1337:
        keygen = default*int(length) +''.join(item_list)
        print(keygen)
                # enable it to save to files.
        #os.system('echo %s >> keygen.txt' % str(keygen))
```


mucomplex@mucomplex-K53SM: ~
File Edit View Search Terminal Help

```
mucomplex@mucomplex-K53SM:~$ python3 keygen.py
value default:q
value length :9
qqqqqqqqqMyz
qqqqqqqqqMzy
qqqqqqqqqNxz
qqqqqqqqqNzx
qqqqqqqqqOwz
qqqqqqqqqOxy
qqqqqqqqqOyx
qqqqqqqqqOzw
qqqqqqqqqPvz
qqqqqqqqqPwy
qqqqqqqqqPyw
qqqqqqqqqPzv
qqqqqqqqqQuz
qqqqqqqqqQvy
qqqqqqqqqQwx
qqqqqqqqqQxw
qqqqqqqqqQyv
qqqqqqqqqQzu
qqqqqqqqqRtz
qqqqqqqqqRuy
qqqqqqqqqRvx
qqqqqqqqqRxv
qqqqqqqqqRyu
qqqqqqqqqRzt
qqqqqqqqqSsz
```

Thanks , have a nice day . Regard **mucomplex** from **3l173Gh057**