

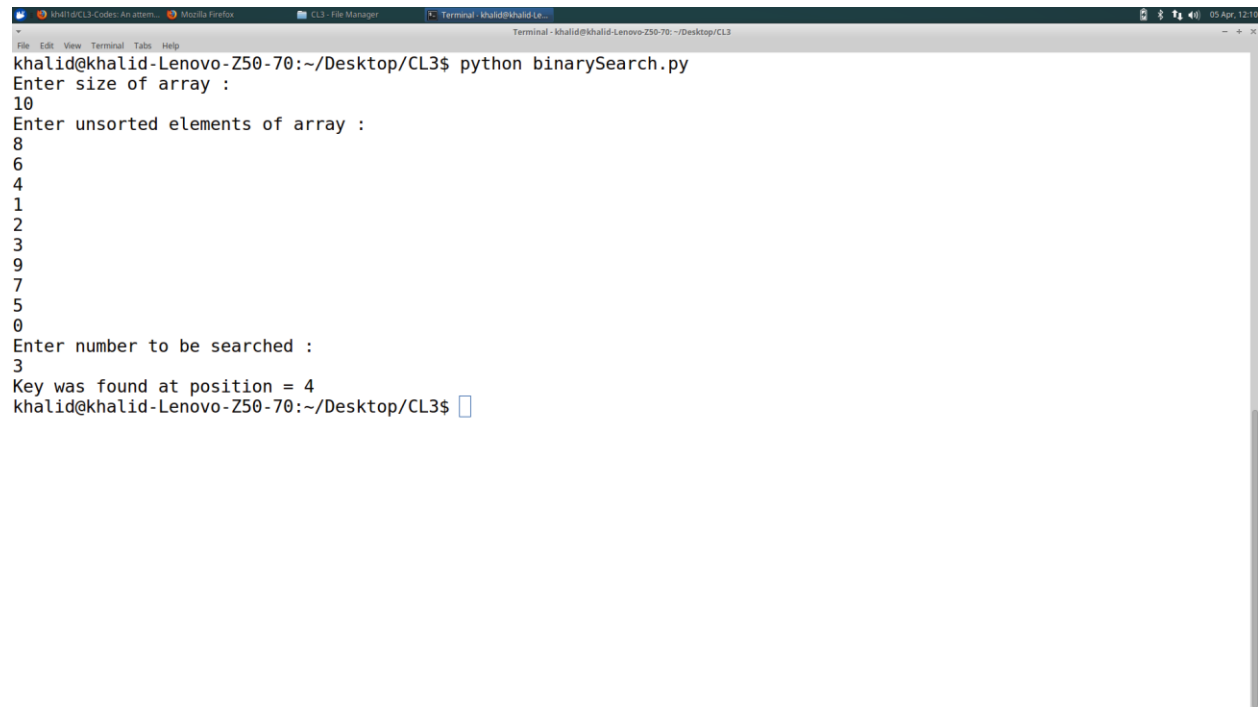
OUTPUT:

```
khalid@khalid-Lenovo-Z50-70:~/Desktop/CL3$ python 8queens.py
[1, 0, 0, 0, 0, 0, 0, 0]
[1, 0, 0, 0, 0, 0, 0, 0]
[1, 0, 0, 0, 0, 0, 0, 0]
[1, 0, 0, 0, 0, 0, 0, 0]
[1, 0, 0, 0, 0, 0, 0, 1]
[1, 0, 0, 0, 0, 0, 0, 0]
[1, 0, 0, 0, 0, 0, 0, 0]
[1, 0, 0, 0, 0, 0, 0, 0]

[7, 1, 3, 0, 6, 4, 2, 5]

[0, 0, 0, 1, 0, 0, 0, 0]
[0, 1, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 1, 0]
[0, 0, 1, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 1, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 1]
[0, 0, 0, 0, 1, 0, 0, 0]
[1, 0, 0, 0, 0, 0, 0, 0]
khalid@khalid-Lenovo-Z50-70:~/Desktop/CL3$
```

OUTPUT:



```
khalid@khalid-Lenovo-Z50-70:~/Desktop/CL3$ python binarySearch.py
Enter size of array :
10
Enter unsorted elements of array :
8
6
4
1
2
3
9
7
5
0
Enter number to be searched :
3
Key was found at position = 4
khalid@khalid-Lenovo-Z50-70:~/Desktop/CL3$
```

The screenshot shows a terminal window titled "Terminal - khalid@khalid-Lenovo-Z50-70: ~/Desktop/CL3". The window contains the output of a Python script named "binarySearch.py". The script prompts the user for the size of an array (10) and then for 10 unsorted elements (8, 6, 4, 1, 2, 3, 9, 7, 5, 0). It then prompts for a number to be searched (3) and outputs "Key was found at position = 4". The terminal window has a menu bar with "File", "Edit", "View", "Terminal", "Tabs", and "Help". The system tray at the top right shows the date and time as "05 Apr, 12:10".

OUTPUT:

```
Terminal - khalid@khalid-Lenovo-Z50-70: ~/Desktop/CL3/Booths
khalid@khalid-Lenovo-Z50-70:~/Desktop/CL3/Booths$ python client.py

Enter the multiplicand :
64

Enter the multiplier :
48

Enter the multiplicand's bit form's length :
10

Enter the multiplier's bit form's length :
10

khalid@khalid-Lenovo-Z50-70:~/Desktop/CL3/Booths$
```

```
Terminal - khalid@khalid-Lenovo-Z50-70: ~/Desktop/CL3/Booths
khalid@khalid-Lenovo-Z50-70:~/Desktop/CL3/Booths$ python server.py
Initial values
A 00010000000000000000
S 11110000000000000000
P 00000000000000110000
Starting calculation
P >> 1: 00000000000000110000
P >> 1: 00000000000000011000
P >> 1: 00000000000000001100
P >> 1: 00000000000000000110
P + S: 11110000000000000110
P >> 1: 11111000000000000011
P >> 1: 11111100000000000001
P + A: 00001100000000000001
P >> 1: 00000110000000000000
P >> 1: 00000011000000000000
P >> 1: 00000001100000000000
P >> 1: 00000000110000000000
P >> 1: 00000000011000000000

The product of 64 and 48 is = 3072

khalid@khalid-Lenovo-Z50-70:~/Desktop/CL3/Booths$
```

OUTPUT:

```
khalid@khalid-Lenovo-Z50-70:~/Desktop/CL3$ python diningPhilosophers.py
p[0] took c[0]
p[1] took c[1]
p[2] took c[2]
p[3] took c[3]
p[3] took c[4]
p[3] i.e. Hume dropped c[4] and thinks -> Donuts exist because I imagine donuts.
p[3] i.e. Hume dropped c[3] and thinks -> Donuts exist because I imagine donuts.
p[2] took c[3]
p[2] i.e. Aristotle dropped c[3] and thinks -> A donut contains it's donut-ness.
p[3] took c[3]
p[2] i.e. Aristotle dropped c[2] and thinks -> A donut contains it's donut-ness.
p[1] took c[2]
p[3] took c[4]
p[1] i.e. Marx dropped c[2] and thinks -> Everybody desires donuts.
p[1] i.e. Marx dropped c[1] and thinks -> Everybody desires donuts.
p[0] took c[1]
p[3] i.e. Hume dropped c[4] and thinks -> Donuts exist because I imagine donuts.
p[4] took c[4]
p[3] i.e. Hume dropped c[3] and thinks -> Donuts exist because I imagine donuts.
p[0] i.e. Descartes dropped c[1] and thinks -> A donut's hope proves it's existence.
p[1] took c[1]
p[0] i.e. Descartes dropped c[0] and thinks -> A donut's hope proves it's existence.
p[4] took c[0]
p[3] took c[3]
p[1] took c[2]
p[4] i.e. Nietzsche dropped c[0] and thinks -> Stop at nothing to get your donut.
p[0] took c[0]
p[4] i.e. Nietzsche dropped c[4] and thinks -> Stop at nothing to get your donut.
p[3] took c[4]
p[1] i.e. Marx dropped c[2] and thinks -> Everybody desires donuts.
p[1] i.e. Marx dropped c[1] and thinks -> Everybody desires donuts.
```

```
Terminal - khalid@khalid-Lenovo-Z50-70: ~/Desktop/CL3
p[3] i.e. Hume dropped c[4] and thinks -> Donuts exist because I imagine donuts.
p[4] took c[4]
p[3] i.e. Hume dropped c[3] and thinks -> Donuts exist because I imagine donuts.
p[2] took c[3]
p[0] i.e. Descartes dropped c[1] and thinks -> A donut's hope proves it's existence.
p[0] i.e. Descartes dropped c[0] and thinks -> A donut's hope proves it's existence.
p[4] took c[0]
p[1] took c[1]
p[2] i.e. Aristotle dropped c[3] and thinks -> A donut contains it's donut-ness.
p[2] i.e. Aristotle dropped c[2] and thinks -> A donut contains it's donut-ness.
p[4] i.e. Nietzsche dropped c[0] and thinks -> Stop at nothing to get your donut.
p[0] took c[0]
p[1] took c[2]
p[4] i.e. Nietzsche dropped c[4] and thinks -> Stop at nothing to get your donut.
p[4] took c[4]
p[1] i.e. Marx dropped c[2] and thinks -> Everybody desires donuts.
p[2] took c[2]
p[1] i.e. Marx dropped c[1] and thinks -> Everybody desires donuts.
p[0] took c[1]
p[2] took c[3]
p[0] i.e. Descartes dropped c[1] and thinks -> A donut's hope proves it's existence.
p[1] took c[1]
p[0] i.e. Descartes dropped c[0] and thinks -> A donut's hope proves it's existence.
p[4] took c[0]
p[2] i.e. Aristotle dropped c[3] and thinks -> A donut contains it's donut-ness.
p[2] i.e. Aristotle dropped c[2] and thinks -> A donut contains it's donut-ness.
p[1] took c[2]
p[4] i.e. Nietzsche dropped c[0] and thinks -> Stop at nothing to get your donut.
p[0] took c[0]
p[4] i.e. Nietzsche dropped c[4] and thinks -> Stop at nothing to get your donut.
p[1] i.e. Marx dropped c[2] and thinks -> Everybody desires donuts.
p[2] took c[2]

p[2] took c[2]
p[1] i.e. Marx dropped c[1] and thinks -> Everybody desires donuts.
p[1] finished thinking and eating
p[2] took c[3]
p[2] i.e. Aristotle dropped c[3] and thinks -> A donut contains it's donut-ness.
p[2] i.e. Aristotle dropped c[2] and thinks -> A donut contains it's donut-ness.
p[2] took c[2]
p[2] took c[3]
p[2] i.e. Aristotle dropped c[3] and thinks -> A donut contains it's donut-ness.
p[2] i.e. Aristotle dropped c[2] and thinks -> A donut contains it's donut-ness.
p[2] took c[2]
p[2] took c[3]
p[2] i.e. Aristotle dropped c[3] and thinks -> A donut contains it's donut-ness.
p[2] i.e. Aristotle dropped c[2] and thinks -> A donut contains it's donut-ness.
p[2] took c[2]
p[2] took c[3]
p[2] i.e. Aristotle dropped c[3] and thinks -> A donut contains it's donut-ness.
p[2] i.e. Aristotle dropped c[2] and thinks -> A donut contains it's donut-ness.
p[2] took c[2]
p[2] took c[3]
p[2] i.e. Aristotle dropped c[3] and thinks -> A donut contains it's donut-ness.
p[2] i.e. Aristotle dropped c[2] and thinks -> A donut contains it's donut-ness.
p[2] finished thinking and eating
khalid@khalid-Lenovo-Z50-70:~/Desktop/CL3$
```

OUTPUT:

```
khalid@khalid-Lenovo-Z50-70:~/Desktop$ python dsaMillerRabin.py

Enter the parameter tuple (p,q,g) :
Enter value for p :
167686
Enter value for q :
89
Enter value for g :
132

Here's how good the DSA tuple is ->
According to brute-force primality testing is q prime ?
True

According to Miller-Rabin primality testing, q is most probably prime ?
True

Is number of bits of q = 160 ?
False

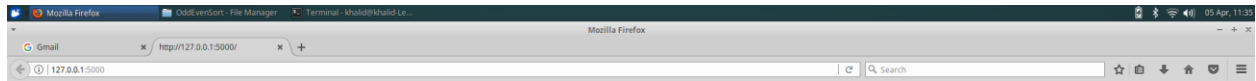
Does q divide (p-1) ?
False

Is number of bits of p between 512 & 1024 ?
False

Is g of the right form i.e.  $(h^{((p-1)/q)} \bmod p)$  where  $h = 2$  ?
False

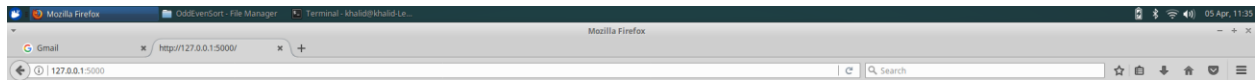
khalid@khalid-Lenovo-Z50-70:~/Desktop$
```

OUTPUT:



Enter elements (separated by ,) to be sorted :

5,4,3,1,2,7,9,6,8 Enter

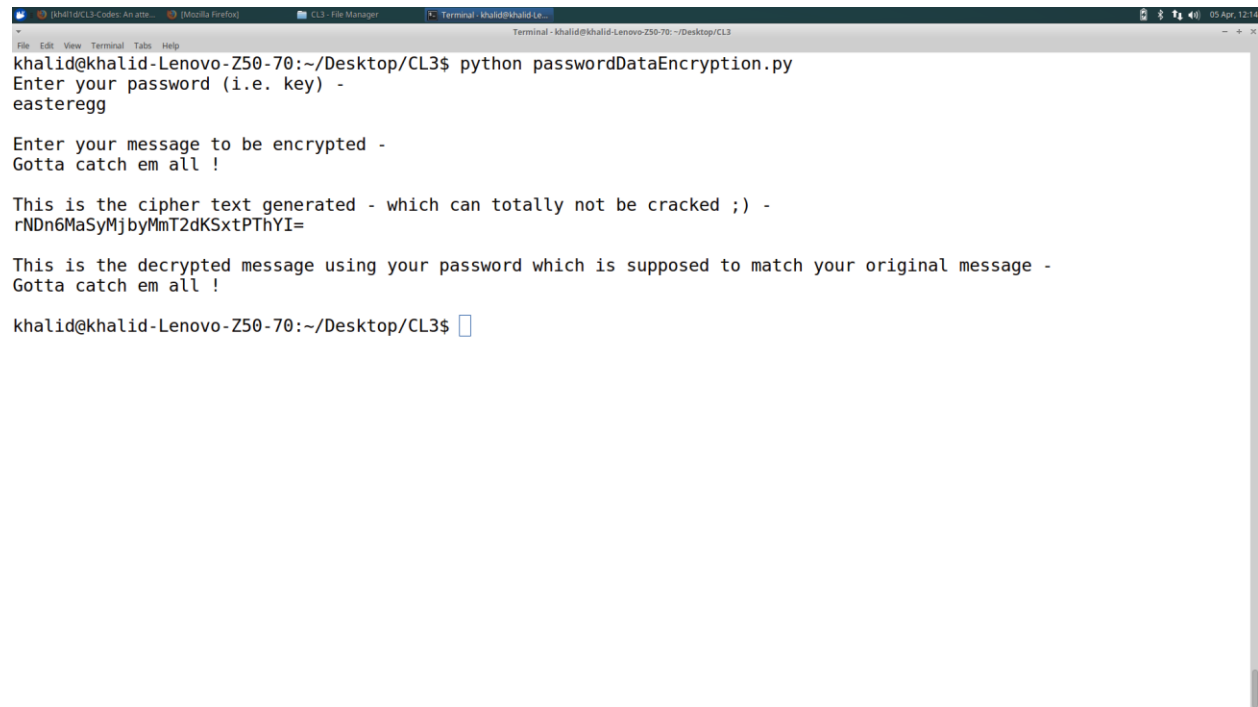


Enter elements (separated by ,) to be sorted :

Enter

[1, 2, 3, 4, 5, 6, 7, 8, 9]

OUTPUT:



```
khalid@khalid-Lenovo-Z50-70:~/Desktop/CL3$ python passwordDataEncryption.py
Enter your password (i.e. key) -
easteregg

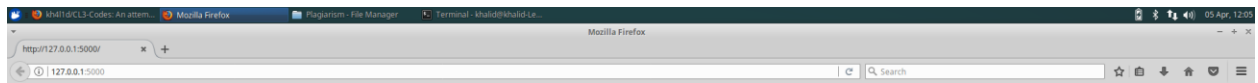
Enter your message to be encrypted -
Gotta catch em all !

This is the cipher text generated - which can totally not be cracked ;) -
rNDn6MaSyMjbyMmt2dKSxtPThYI=

This is the decrypted message using your password which is supposed to match your original message -
Gotta catch em all !

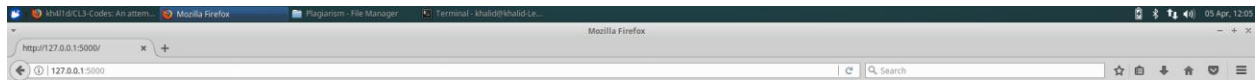
khalid@khalid-Lenovo-Z50-70:~/Desktop/CL3$
```


OUTPUT:



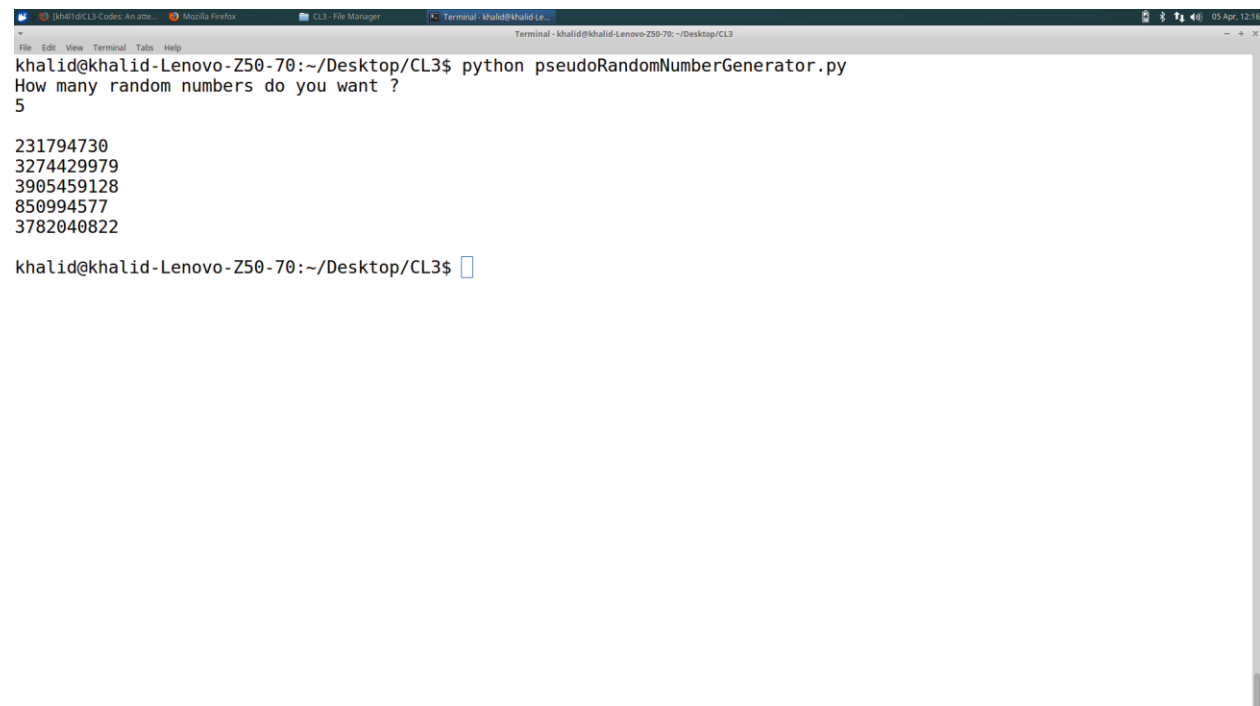
Enter the texts to be compared

the quick brownie ended up: wine ended up in my tummy Check!



Plagiarism Detected !

OUTPUT:

A screenshot of a Linux terminal window. The window title is "Terminal - khalid@khalid-Lenovo-Z50-70: ~/Desktop/CL3". The terminal shows the command "python pseudoRandomNumberGenerator.py" being executed. The prompt asks "How many random numbers do you want ?" and the user enters "5". The script then outputs five random numbers: 231794730, 3274429979, 3905459128, 850994577, and 3782040822. The prompt "khalid@khalid-Lenovo-Z50-70:~/Desktop/CL3\$" is visible at the bottom.

```
khalid@khalid-Lenovo-Z50-70:~/Desktop/CL3$ python pseudoRandomNumberGenerator.py
How many random numbers do you want ?
5

231794730
3274429979
3905459128
850994577
3782040822

khalid@khalid-Lenovo-Z50-70:~/Desktop/CL3$
```

OUTPUT:

```
khalid@khalid-Lenovo-Z50-70:~/Desktop/CL3$ python createXMLfile.py
Enter the name of file: items
items.xml file is written & press 1 to see its contents:1
<Numbers>
  <integer num = "645" ></integer>
  <integer num = "715" ></integer>
  <integer num = "172" ></integer>
  <integer num = "518" ></integer>
  <integer num = "387" ></integer>
  <integer num = "465" ></integer>
  <integer num = "442" ></integer>
  <integer num = "458" ></integer>
  <integer num = "86" ></integer>
  <integer num = "58" ></integer>
</Numbers>

press 2 to start quickstart program: 2
[58, 86, 172, 387, 442, 458, 465, 518, 645, 715]
khalid@khalid-Lenovo-Z50-70:~/Desktop/CL3$
```

OUTPUT:



The image displays two screenshots of a terminal window. The top screenshot shows the execution of a Python script named 'client.py'. The prompt is 'khalid@khalid-Lenovo-Z50-70:~/Desktop/CL3/SHA-1\$'. The user enters the password 'tukkupaglu' to gain a Kerberos ticket. The bottom screenshot shows the execution of a Python script named 'server.py'. The prompt is 'khalid@khalid-Lenovo-Z50-70:~/Desktop/CL3/SHA-1\$'. The user enters the password 'tukkupaglu' as the password registered with the Kerberos Server. The server responds with 'Kerberos ticket granted !'.

```
khalid@khalid-Lenovo-Z50-70:~/Desktop/CL3/SHA-1$ python client.py

Enter your password to gain a Kerberos ticket :
tukkupaglu

khalid@khalid-Lenovo-Z50-70:~/Desktop/CL3/SHA-1$
```



```
khalid@khalid-Lenovo-Z50-70:~/Desktop/CL3/SHA-1$ python server.py

Enter password registered with Kerberos Server :
tukkupaglu

Kerberos ticket granted !

khalid@khalid-Lenovo-Z50-70:~/Desktop/CL3/SHA-1$
```

OUTPUT:

