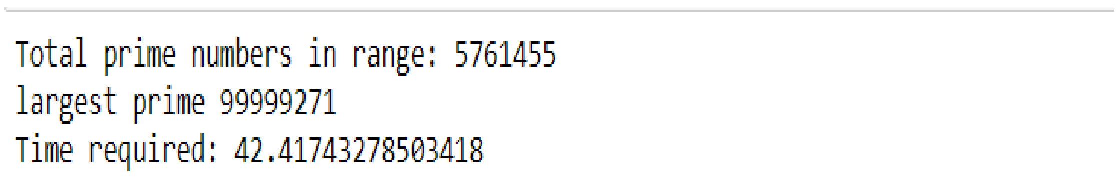
CB.EN.P2CSE19002
AKSHAYA R

BIG DATA SECURITY

1. Code for Finding a large prime number (min 20-digit) (Paste code and screenshots of output)

```
import time
def SieveOfEratosthenes(n):
    prime = [True for i in range(n+1)]
    p = 2
    while(p * p <= n):
        if (prime[p] == True):
            for i in range(p * p, n + 1, p):
                prime[i] = False
        p += 1
        c = 0
    for p in range(2, n):
        if prime[p]:
            c += 1
    return c,i
t0 = time.time()
c,i = SieveOfEratosthenes(100000000)
print("Total prime numbers in range:", c )
print("largest prime", i)
t1 = time.time()
print("Time required:", t1 - t0)
```

OUTPUT:

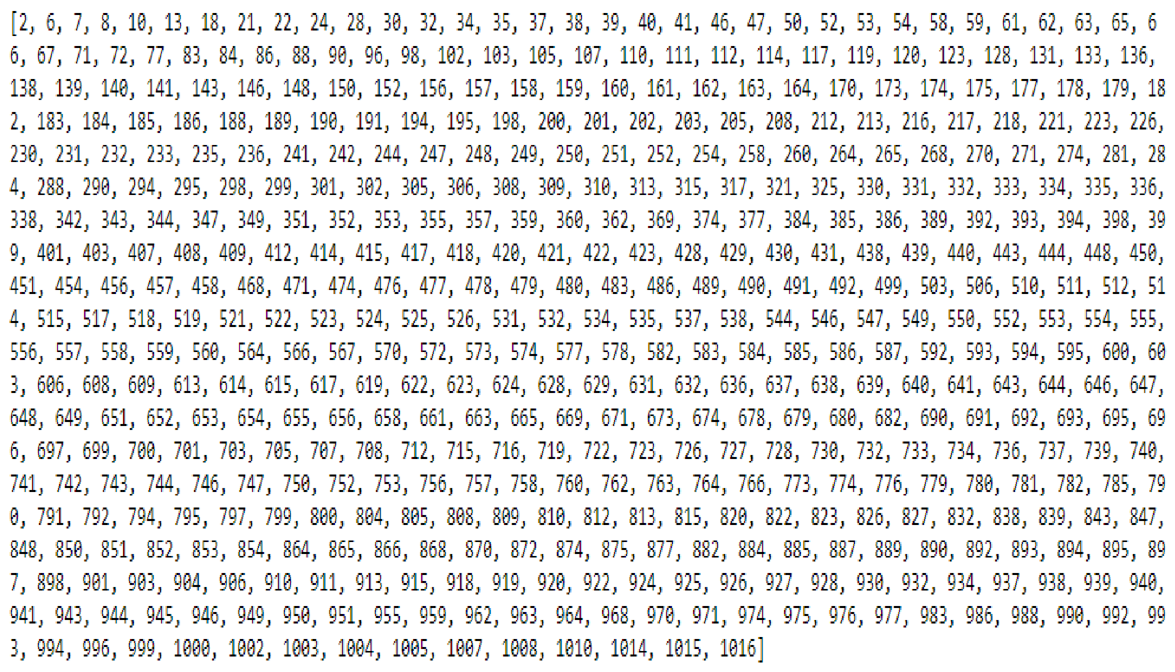


```
Total prime numbers in range: 5761455
largest prime 99999271
Time required: 42.41743278503418
```

2. Code for finding Primitive root (min 10 digits) (Paste code and screenshots of output)

```
from math import gcd as bltin_gcd
def primRoots(modulo):
    required_set = {num for num in range(1, modulo) if bltin_gcd(num, modulo) }
    return [g for g in range(1, modulo) if required_set == {pow(g, powers, modulo)
            for powers in range(1, modulo)}]
print(primRoots(1019))
```

OUTPUT:



```
[2, 6, 7, 8, 10, 13, 18, 21, 22, 24, 28, 30, 32, 34, 35, 37, 38, 39, 40, 41, 46, 47, 50, 52, 53, 54, 58, 59, 61, 62, 63, 65, 6
6, 67, 71, 72, 77, 83, 84, 86, 88, 90, 96, 98, 102, 103, 105, 107, 110, 111, 112, 114, 117, 119, 120, 123, 128, 131, 133, 136,
138, 139, 140, 141, 143, 146, 148, 150, 152, 156, 157, 158, 159, 160, 161, 162, 163, 164, 170, 173, 174, 175, 177, 178, 179, 18
2, 183, 184, 185, 186, 188, 189, 190, 191, 194, 195, 198, 200, 201, 202, 203, 205, 208, 212, 213, 216, 217, 218, 221, 223, 226,
230, 231, 232, 233, 235, 236, 241, 242, 244, 247, 248, 249, 250, 251, 252, 254, 258, 260, 264, 265, 268, 270, 271, 274, 281, 28
4, 288, 290, 294, 295, 298, 299, 301, 302, 305, 306, 308, 309, 310, 313, 315, 317, 321, 325, 330, 331, 332, 333, 334, 335, 336,
338, 342, 343, 344, 347, 349, 351, 352, 353, 355, 357, 359, 360, 362, 369, 374, 377, 384, 385, 386, 389, 392, 393, 394, 398, 39
9, 401, 403, 407, 408, 409, 412, 414, 415, 417, 418, 420, 421, 422, 423, 428, 429, 430, 431, 438, 439, 440, 443, 444, 448, 450,
451, 454, 456, 457, 458, 468, 471, 474, 476, 477, 478, 479, 480, 483, 486, 489, 490, 491, 492, 499, 503, 506, 510, 511, 512, 51
4, 515, 517, 518, 519, 521, 522, 523, 524, 525, 526, 531, 532, 534, 535, 537, 538, 544, 546, 547, 549, 550, 552, 553, 554, 555,
556, 557, 558, 559, 560, 564, 566, 567, 570, 572, 573, 574, 577, 578, 582, 583, 584, 585, 586, 587, 592, 593, 594, 595, 600, 60
3, 606, 608, 609, 613, 614, 615, 617, 619, 622, 623, 624, 628, 629, 631, 632, 636, 637, 638, 639, 640, 641, 643, 644, 646, 647,
648, 649, 651, 652, 653, 654, 655, 656, 658, 661, 663, 665, 669, 671, 673, 674, 678, 679, 680, 682, 690, 691, 692, 693, 695, 69
6, 697, 699, 700, 701, 703, 705, 707, 708, 712, 715, 716, 719, 722, 723, 726, 727, 728, 730, 732, 733, 734, 736, 737, 739, 740,
741, 742, 743, 744, 746, 747, 750, 752, 753, 756, 757, 758, 760, 762, 763, 764, 766, 773, 774, 776, 779, 780, 781, 782, 785, 79
0, 791, 792, 794, 795, 797, 799, 800, 804, 805, 808, 809, 810, 812, 813, 815, 820, 822, 823, 826, 827, 832, 838, 839, 843, 847,
848, 850, 851, 852, 853, 854, 864, 865, 866, 868, 870, 872, 874, 875, 877, 882, 884, 885, 887, 889, 890, 892, 893, 894, 895, 89
7, 898, 901, 903, 904, 906, 910, 911, 913, 915, 918, 919, 920, 922, 924, 925, 926, 927, 928, 930, 932, 934, 937, 938, 939, 940,
941, 943, 944, 945, 946, 949, 950, 951, 955, 959, 962, 963, 964, 968, 970, 971, 974, 975, 976, 977, 983, 986, 988, 990, 992, 99
3, 994, 996, 999, 1000, 1002, 1003, 1004, 1005, 1007, 1008, 1010, 1014, 1015, 1016]
```

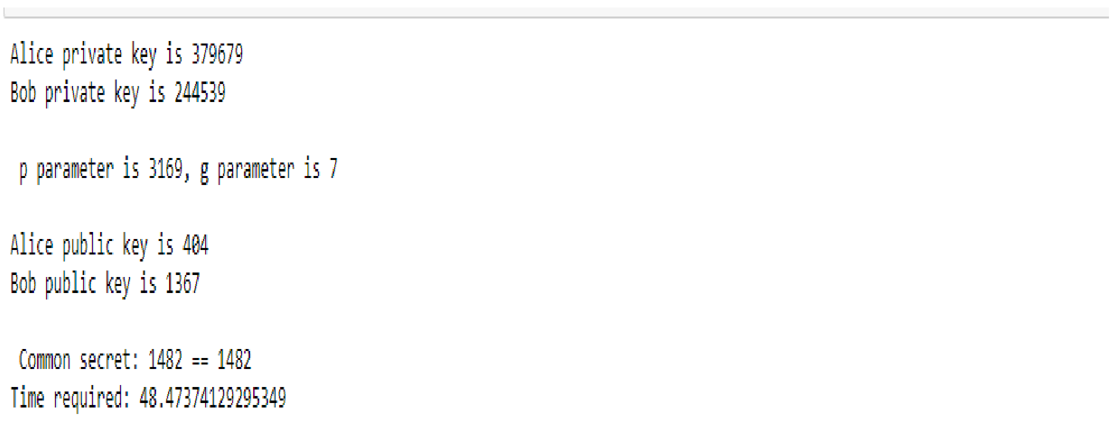
3. Code for DH (Taking random private keys (Min 3 of different sizes with min 8 digits) (Paste code and output screenshot)

```
from random import getrandbits
from random import randint
import sys
```

```
def is_prime_calc(num):
    return all(num % i for i in range(2, num))

def is_prime(num):
    return is_prime_calc(num)

def get_random_prime():
    while True:
        n = getrandbits(12) + 3;
        if is_prime(n):
            return n

def gcd(a,b):
    while a != b:
        if a > b:
            a = a - b
        else:
            b = b - a
    return a

def primitive_root(modulo):
    required_set = set(num for num in range (1, modulo) if gcd(num, modulo) == 1)
    for g in range(1, modulo):
        actual_set = set(pow(g, powers) % modulo for powers in range (1, modulo))
        if required_set == actual_set:
            return g

# Generating private keys
alice_private = randint(999, 999999)
print ('Alice private key is %d' % alice_private)
bob_private = randint(999, 999999)
```

```
print ('Bob private key is %d' % bob_private)

# Generating p-g parameters
p = get_random_prime()
g = primitive_root(p)

print ('\n p parameter is %d, g parameter is %d \n' % (p, g))

# Generating public keys
alice_public = pow(g, alice_private) % p
bob_public = pow(g, bob_private) % p

print ('Alice public key is %d' % alice_public)
print ('Bob public key is %d' % bob_public)

alice_key = (pow(bob_public, alice_private)) % p
bob_key = (pow(alice_public, bob_private)) % p

print ('\n Common secret: %d == %d' % (alice_key, bob_key))
print("Time required:", t1 - t0)
```

OUTPUT:



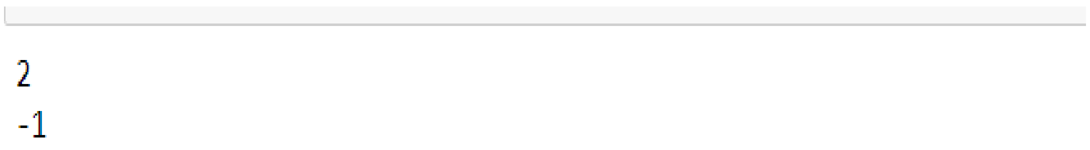
```
Alice private key is 379679
Bob private key is 244539

 p parameter is 3169, g parameter is 7

Alice public key is 404
Bob public key is 1367

 Common secret: 1482 == 1482
Time required: 48.47374129295349
```

4. Code to Find an integer k such that $a^k \equiv b \pmod{m}$ where a and m are relatively prime.(Paste code and output)

```
import math;

def discreteLogarithm(a, b, m):

    n = int(math.sqrt (m) + 1);

    # Calculate a ^ n
    an = 1;
    for i in range(n):
        an = (an * a) % m;

    value = [0] * m;

    # Store all values of a^(n*i) of LHS
    cur = an;
    for i in range(1, n + 1):
        if (value[ cur ] == 0):
            value[ cur ] = i;
        cur = (cur * an) % m;

    cur = b;
    for i in range(n + 1):

        # Calculate (a ^ j) * b and check
        # for collision
        if (value[cur] > 0):
            ans = value[cur] * n - i;
            if (ans < m):
                return ans;
        cur = (cur * a) % m;

    return -1;

# Driver code
a = 200;
b = 3;
m = 5;
print(discreteLogarithm(a, b, m));

a = 350;
```

```
b = 7;
m = 11;
print(discreteLogarithm(a, b, m));
```

OUTPUT:

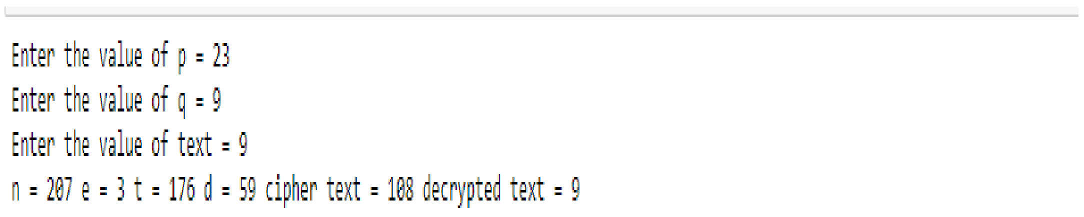


```
2
-1
```

Question 5

```
from decimal import Decimal


def gcd(a,b):
    if b==0:
        return a
    else:
        return gcd(b,a%b)
p = int(input('Enter the value of p = '))
q = int(input('Enter the value of q = '))
no = int(input('Enter the value of text = '))
n = p*q
t = (p-1)*(q-1)


for e in range(2,t):
    if gcd(e,t)== 1:
        break


for i in range(1,10):
    x = 1 + i*t
    if x % e == 0:
        d = int(x/e)
        break
ctt = Decimal(0)
ctt =pow(no,e)
ct = ctt % n

dtt = Decimal(0)
dtt = pow(ct,d)
dt = dtt % n

print('n = '+str(n)+' e = '+str(e)+' t = '+str(t)+' d = '+str(d)+' cipher text = '+str(ct)+' decrypted text = '+str(dt))
```

OUTPUT:

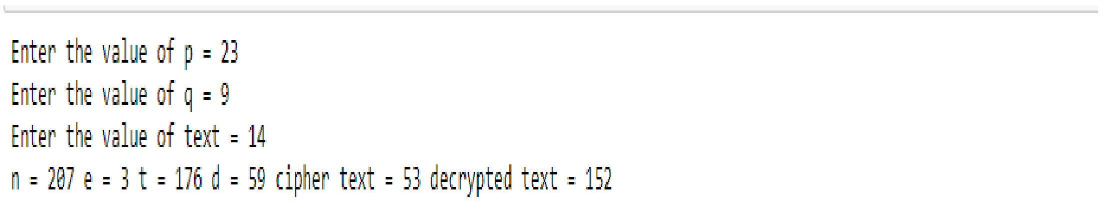


```
Enter the value of p = 23
Enter the value of q = 9
Enter the value of text = 9
n = 207 e = 3 t = 176 d = 59 cipher text = 108 decrypted text = 9
```

```
from decimal import Decimal
def gcd(a,b):
    if b==0:
        return a
    else:
        return gcd(b,a%b)
p = int(input('Enter the value of p = '))
q = int(input('Enter the value of q = '))
no = int(input('Enter the value of text = '))
n = p*q
```

```
t = (p-1)*(q-1)

for e in range(2,t):
    if gcd(e,t)== 1:
        break
for i in range(1,10):
    x = 1 + i*t
    if x % e == 0:
        d = int(x/e)
        break
ctt = Decimal(0)
ctt =pow(no,e)
ct = ctt % n

dtt = Decimal(0)
dtt = pow(ct,d)
dt = dtt % n

print('n = '+str(n)+' e = '+str(e)+' t = '+str(t)+' d = '+str(d)+' cipher text = '+str(ct)+' decrypted text = '+str(dt))
```

OUTPUT:

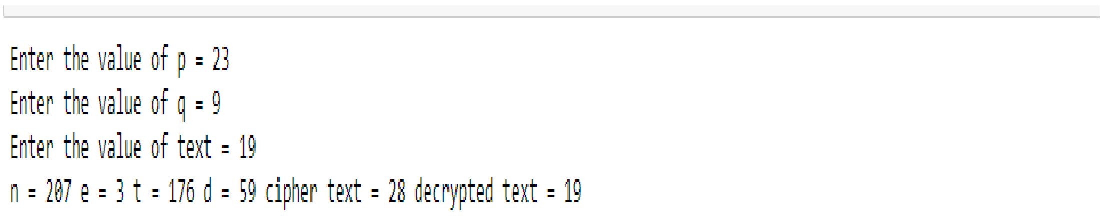


```
Enter the value of p = 23
Enter the value of q = 9
Enter the value of text = 14
n = 207 e = 3 t = 176 d = 59 cipher text = 53 decrypted text = 152
```

```
from decimal import Decimal
def gcd(a,b):
    if b==0:
        return a
```

```
 else:
        return gcd(b,a%b)
p = int(input('Enter the value of p = '))
q = int(input('Enter the value of q = '))
no = int(input('Enter the value of text = '))
n = p*q
t = (p-1)*(q-1)

for e in range(2,t):
    if gcd(e,t)== 1:
        break
for i in range(1,10):
    x = 1 + i*t
    if x % e == 0:
        d = int(x/e)
        break
ctt = Decimal(0)
ctt =pow(no,e)
ct = ctt % n
dtt = Decimal(0)
dtt = pow(ct,d)
dt = dtt % n
print('n = '+str(n)+' e = '+str(e)+' t = '+str(t)+' d = '+str(d)+' cipher text = '+str(ct)+' decrypted
text = '+str(dt))
```

OUTPUT:



```
Enter the value of p = 23
Enter the value of q = 9
Enter the value of text = 19
n = 207 e = 3 t = 176 d = 59 cipher text = 28 decrypted text = 19
```