In [ ]:

```
#Q1 Largest prime number
```

In [10]:

```
from random import randrange, getrandbits
def is_prime(n, k=128):
    if n == 2 or n == 3:
        return True
    if n <= 1 or n % 2 == 0:
        return False
    s = 0
    r = n - 1
    while r & 1 == 0:
        s += 1
        r //= 2
    for _ in range(k):
        a = randrange(2, n - 1)
        x = pow(a, r, n)
        if x != 1 and x != n - 1:
            j = 1
            while j < s and x != n - 1:
                x = pow(x, 2, n)
                if x == 1:
                    return False
                j += 1
            if x != n - 1:
                return False
    return True
def generate_prime_candidate(length):
    p = getrandbits(length)
    p |= (1 << length - 1) | 1
    return p
def generate_prime_number(length=100):
    p = 4
    while not is_prime(p, 128):
        p = generate_prime_candidate(length)
    return p
print(generate_prime_number())
```

```
1046510494338055519355002077709
```

In [ ]:

```
#Q2 primitive roots
```

In [13]:

```
from math import sqrt
def isPrime( n):
    if (n <= 1):
        return False
    if (n <= 3):
        return True
    if (n % 2 == 0 or n % 3 == 0):
        return False
    i = 5
    while(i * i <= n):
        if (n % i == 0 or n % (i + 2) == 0) :
            return False
        i = i + 6

    return True
def power( x, y, p):
    res = 1
    x = x % p
    while (y > 0):
        if (y & 1):
            res = (res * x) % p
        y = y >> 1
        x = (x * x) % p
    return res

def findPrimefactors(s, n) :
    while (n % 2 == 0) :
        s.add(2)
        n = n // 2
    for i in range(3, int(sqrt(n)), 2):
        while (n % i == 0) :

            s.add(i)
            n = n // i
    if (n > 2) :
        s.add(n)

def findPrimitive( n) :
    s = set()

    if (isPrime(n) == False):
        return -1
    phi = n - 1
    findPrimefactors(s, phi)
    for r in range(2, phi + 1):
        flag = False
        for it in s:
            if (power(r, phi // it, n) == 1):

                flag = True
                break
        if (flag == False):
            return r
    return -1
n = 5159481525

print("Smallest primitive root of",n, "is", findPrimitive(n))
from math import gcd
def countPrimitiveRoots(p):
    result = 1
    for i in range(2, p, 1):
        if (gcd(i, p) == 1):
            result += 5159481525

    return result

if __name__ == '__main__':
    p = 5159481525
```

```
Smallest primitive root of 5159481525 is -1
```

In [14]:

```
#Q3 DH
```

In [27]:

```
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
alice_private = randint(222, 222222)
print ('Alice private key is %d' % alice_private)
bob_private = randint(222, 222222)
print ('bob private key is %d' % bob_private)
p = get_random_prime()
g = primitive_root(p)
print ('\n p parameter is %d, g parameter is %d \n' % (p, g))
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(alice_public, alice_private)) % p
bob_key = (pow(bob_public, bob_private)) % p

print ('\n Common secret: %d == %d' % (alice_key, bob_key))
```

```
Alice private key is 194784
bob private key is 36321

 p parameter is 3631, g parameter is 15

Alice public key is 1639
bob public key is 3494

 Common secret: 2473 == 1030
```

In [17]:

```
#Q4 Find k
```

In [35]:

```
import math;

def discreteLogarithm(a, b, m):

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

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

    value = [0] * m;

    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):

        if (value[cur] > 0):
            ans = value[cur] * n - i;
            if (ans < m):
                return ans;
        cur = (cur * a) % m;

    return -1;
a = 5;
b = 7;
m = 9;
print(discreteLogarithm(a, b, m));

a = 23;
b = 29;
m = 31;
print(discreteLogarithm(a, b, m));
```

```
8
17
```

Q5

In [ ]:

```
#1
```

In [37]:

```
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 = '))
g = int(input('Enter the value of g = '))
no = int(input('Enter the value of text = '))
n = p*g
t = (p-1)*(g-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)+
```

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

In [38]:

```
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)+
```

```
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
```

In [39]:

```
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)+
```

```
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
```

In [43]:

```

```

In [ ]:

```

```