

HOJA DE EJERCICIOS 9 – ALGORITMOS AVANZADOS UNSAAC

Ejercicio 1.

Efectúa un análisis similar para $p = 29$, $q = 31$. Muestra los 5 pasos del algoritmo, los valores obtenidos, las claves públicas y privadas, y las funciones de encriptación y descriptación.

① $p = 29$, $q = 31$

- 1= Los números primos son: 29, 31
- 2= Calculamos $n = p \times q = 29 \times 31 = 899$
- 3= Calculamos el totiente:
$$\phi(n) = (p-1)(q-1)$$
$$\phi(899) = (29-1)(31-1) = 840$$
- 4= Escogemos $e > 1$, co-primo con 840.
 $e = 31$
- 5= Escogemos d , que satisfaga $de \equiv 1 \pmod{\phi(n)} \Rightarrow d(31) \equiv 1 \pmod{840}$
 $\Rightarrow d = 271$
Porque $31 \times 271 = 8401 = 1 + 10 \times 840$

Entonces:

- La clave pública es $(n=899, e=31)$, Para una Mensaje m la función de encriptación, es:
$$C = m^e \pmod{n}$$
- La clave privada es $(n=899, d=271)$, La función de descriptación es:
$$m = C^d \pmod{n}$$

Ejercicio 2.

Toma otros dos números primos menores de 100 diferentes al ejercicio visto en clase, y a los dos anteriores:

2	3	5	7	11
13	17	19	23	29
31	37	41	43	47
53	59	61	67	71
73	79	83	89	97

Y efectúa un análisis similar. Muestra los 5 pasos del algoritmo, los valores obtenidos, las claves públicas y privadas, y las funciones de encriptación y desencriptación.

②

1= Escogemos los números: $p=3$ y $q=7$

2= Calculamos $n=p \times q = 21$

3= Calculamos el totiente $\phi(n) = (p-1)(q-1)$

$$\phi(21) = \phi(21) = (3-1)(7-1) = 2 \times 6 = 12$$

4= Escogemos $e > 1$, co-primo con 12

$$e=5$$

5= Escogemos d , que satisfaga $de \equiv 1 \pmod{\phi(n)}$

$$d=5$$

$$\text{Porque } 5 \times 5 = 25 = 1 + 2 \times 12$$

Entonces:

- La clave pública es $(n=21, e=5)$. Para un mensaje m la función de encriptación, es:

$$C = m^e \pmod{n}$$

- La clave privada es $(n=21, d=5)$, la función de desencriptación es:

$$m = C^d \pmod{n}$$

Ejercicio 3.

Revisa el siguiente código implementado en Python:

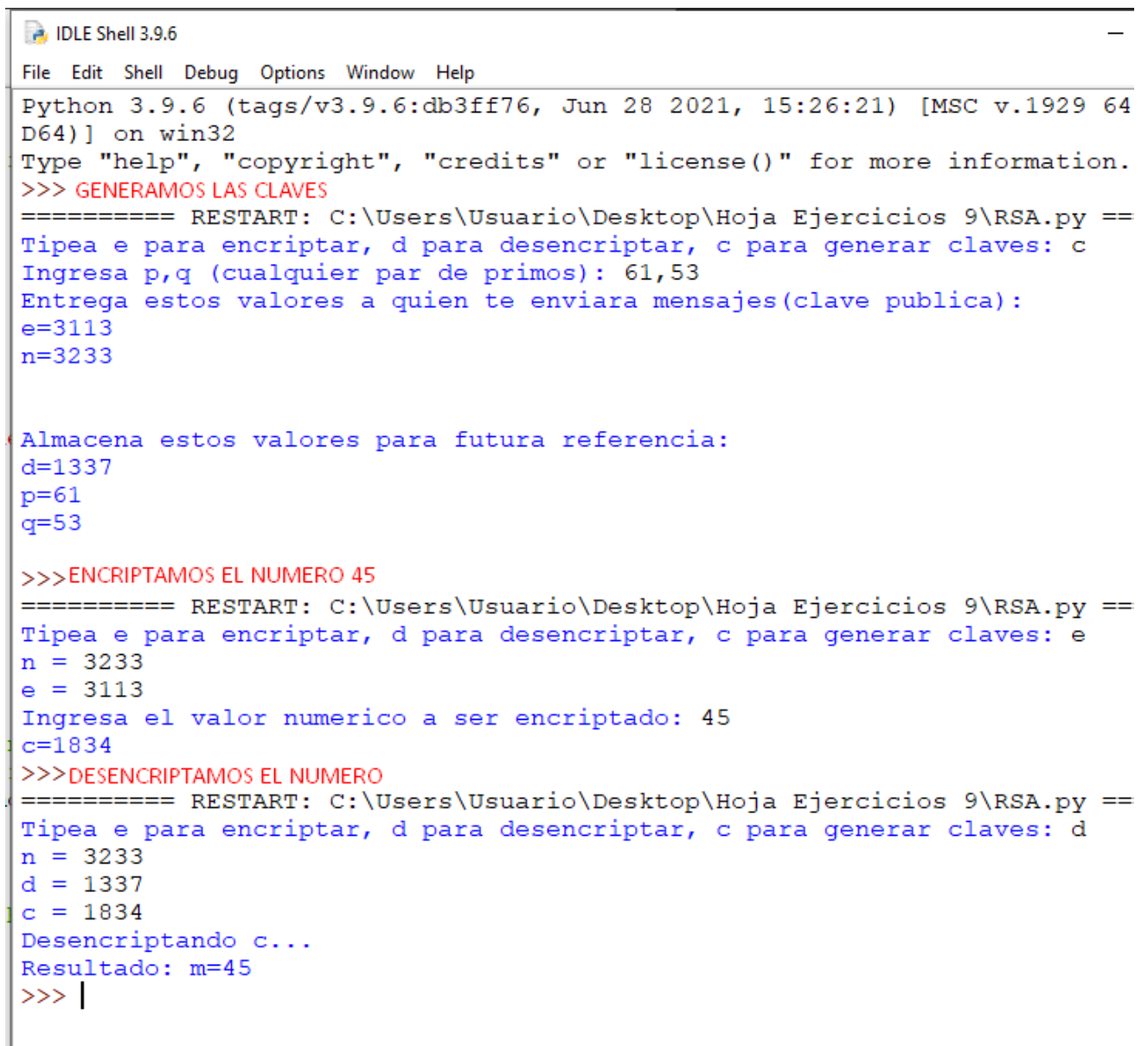
```
from math import gcd

def phi(a,b):
    return (a-1)*(b-1)

def coprime(n):
    # Encontremos el valor mas grande co-primo con n
    j=1
    for i in range(2, (n-1)):
        if gcd(n,i) == 1:
            j = i
    return j

def esprimo(n):
    # aseguramos que n es positivo
    n = abs(int(n))
    # 0 y 1 no son primos
    if n < 2:
        return False
    # 2 es el unico primo par
    if n == 2:
        return True
    # todos los otros numeros pares no son primos
    if not n & 1:
        return False
    # range inicia en 3 y crece la raiz cuadrada de n
    # para todos los numeros impares
    for x in range(3, int(n*0.5)+1, 2):
        if n % x == 0:
```

A. Verifica el correcto funcionamiento para encriptar, desencriptar y generar claves. Muestra capturas de pantalla para cada caso.



```
IDLE Shell 3.9.6
File Edit Shell Debug Options Window Help
Python 3.9.6 (tags/v3.9.6:db3ff76, Jun 28 2021, 15:26:21) [MSC v.1929 64
D64] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> GENERAMOS LAS CLAVES
===== RESTART: C:\Users\Usuario\Desktop\Hoja Ejercicios 9\RSA.py ==
Tipea e para encriptar, d para desencriptar, c para generar claves: c
Ingresa p,q (cualquier par de primos): 61,53
Entrega estos valores a quien te enviara mensajes(clave publica):
e=3113
n=3233

Almacena estos valores para futura referencia:
d=1337
p=61
q=53

>>> ENCRYPTAMOS EL NUMERO 45
===== RESTART: C:\Users\Usuario\Desktop\Hoja Ejercicios 9\RSA.py ==
Tipea e para encriptar, d para desencriptar, c para generar claves: e
n = 3233
e = 3113
Ingresa el valor numerico a ser encriptado: 45
c=1834

>>> DESENCRIPTAMOS EL NUMERO
===== RESTART: C:\Users\Usuario\Desktop\Hoja Ejercicios 9\RSA.py ==
Tipea e para encriptar, d para desencriptar, c para generar claves: d
n = 3233
d = 1337
c = 1834
Desencriptando c...
Resultado: m=45
>>> |
```

B. La función coprimo devuelve el valor co-primo más alto para un n dado. Modifica el código, de tal manera que devuelva aleatoriamente un número co-primo con n, no necesariamente el más alto.

```
def coprimo(n):
    #Encontremos el valor mas grande co-primo con n
    j=1
    for i in range(2, (n-1)):
        if(gcd(n,i)==1):
            j=i
            if(random.randint(1,300)%4==0):
                return j

'''
===== RESTART: C:\Users\Usuario\Desktop\Hoja Ejercicios 9\RSA.py =
Tipea e para encriptar, d para desencriptar, c para generar claves: c
Ingresa p,q (cualquier par de primos): 5,7
Entrega estos valores a quien te enviara mensajes(clave publica):
e=11 ←
n=35

Almacena estos valores para futura referencia:
d=11
p=5
q=7

Tipea e para encriptar, d para desencriptar, c para generar claves: c
Ingresa p,q (cualquier par de primos): 5,7
Entrega estos valores a quien te enviara mensajes(clave publica):
e=7 ←
n=35

Almacena estos valores para futura referencia:
d=7
p=5
q=7

Tipea e para encriptar, d para desencriptar, c para generar claves: c
Ingresa p,q (cualquier par de primos): 5,7
Entrega estos valores a quien te enviara mensajes(clave publica):
e=5 ←
n=35

Almacena estos valores para futura referencia:
d=5
```

C. Al generar claves el programa no valida que ambos números dados (p y q) sean primos. Modifica el programa para que valide que ambos valores sean primos usando la función `esprimo`.

Agregamos los cambios al programa principal

```
if __name__=="__main__":
    while(True):
        b=input("Tipea e para encriptar, d para desencriptar, c para generar claves: ")

        if b=='d':
            n=int(input("n = "))
            d=int(input("d = "))
            c=int(input("c = "))
            desencriptar(n,d,c)
        elif b=='e':
            n=int(input("n = "))
            e=int(input("e = "))
            encriptar(n,e)
        elif b=='c':
            i=tuple(int(x.strip())for x in input("Ingresa p,q (cualquier par de primos): ").split(","))
            #Validamos si los dos numeros son primos
            n1,n2=i
            while(not(esprimo(n1) and esprimo(n2))):
                print("INGRESE NUEVAMENTE LOS NUMEROS(tienen que ser primos)")
                i=tuple(int(x.strip())for x in input("Ingresa p,q (cualquier par de primos): ").split(","))
                n1,n2=i
            generadoresclaves(i)
```

Ejercicio 4.

Observa el video relacionado al Teorema Chino del Resto:

https://www.youtube.com/watch?v=l9dXo5f3zDc&ab_channel=Numberphile

Resume y explica con detalle que se está mostrando (“The last cards match”) y porque tal resultado es posible.

El juego mostrado en el video usa dos mazos de 4 cartas en cada una y cada una de ellas tienen orden distinto, utilizando la palabra **NUMBERPHILE** se obtiene 4 parejas iguales, esto es debido a que se cumple la siguiente ecuación:

$$n \equiv -1 \pmod{k}$$

Donde n toma la cantidad de caracteres que tiene la palabra **NUMBERPHILE**, k es el número de cartas que hay en el mazo (esta ira disminuyendo cada vez que se retire un par de cartas)

$$11 \equiv -1 \pmod{4}$$

$$11 \equiv -1 \pmod{3}$$

$$11 \equiv -1 \pmod{2}$$

$$11 \equiv -1 \pmod{1}$$