UNIVERSIDAD CENTRAL DEL ECUADOR

Facultad de ingeniería y ciencias aplicadas

Carrera: Computación

Asignatura: Criptografía y Seguridad de la Información.

Tarea 01: Resultados.

Estudiantes:

1. Andrade Cardenas Kevin Anthony
2. Chamba Carrion Jordy Pedro
3. Luna Grijalva Joel Joshua
4. Parra Vásquez Rensso Nicolay
5. Pozo Maldonado Kevin Fernando
6. Verkade Montiel Emil Thaddaeus

Ejercicio 01:

Algoritmo:

```python
def permutar(palabra):
    if len(palabra) <= 1:
        return [palabra]

    resultado = []
    for i, letra in enumerate(palabra):
        resto = palabra[:i] + palabra[i+1:]
        for p in permutar(resto):
            resultado.append(letra + p)
    return resultado
```

[4] ✓ 0.0s

Precaución para malas entradas:

```python
def obtener_palabra_valida():
    while True:
        input_palabra = input("Ingrese una palabra: ").strip().lower()

        if not input_palabra:
            print("Error: No ha ingresado ninguna palabra. Intente nuevamente.")
            continue

        if not input_palabra.isalpha():
            print("Error: Solo se permiten caracteres alfabéticos. Intente nuevamente.")
            continue

        return input_palabra
```

[5] ✓ 0.0s

El algoritmo implementado genera todas las permutaciones posibles de una cadena mediante un enfoque recursivo con backtracking. Su operación sigue estos principios fundamentales:

1) **Caso Base**: Cuando la cadena tiene un solo carácter, devuelve una lista conteniendo únicamente esa cadena.
2) **Paso Recursivo**: Para cadenas de longitud $n > 1$:
    a) Selecciona cada carácter como elemento pivote
    b) Genera todas las permutaciones del subconjunto restante (excluyendo el pivote)
    c) Combina el pivote con cada una de estas sub-permutaciones
    d) Almacena los resultados en una lista acumulativa

```python
print("Generador de Permutaciones de Palabras")
print("--------------------------------------")

try:
    palabra = obtener_palabra_valida()
    permutaciones = sorted(set(permutar(palabra)))  # Elimina duplicados y ordena

    print(f"\nTotal permutaciones únicas: {len(permutaciones)}")
    print("\nPrimeras 10 permutaciones:")

    for i, p in enumerate(permutaciones[:10], 1):
        print(f"{i}. {p}")

except Exception as e:
    print(f"\nOcurrió un error inesperado: {e}")
```

[6] ✓ 5.1s

Finalmente se ejecuta y se aplica el detalle de imprimir los primeros 10 en orden alfabético.

Resultado:

```
Generador de Permutaciones de Palabras
--------------------------------------
Error: No ha ingresado ninguna palabra. Intente nuevamente.

Total permutaciones únicas: 24

Primeras 10 permutaciones:
1. ahlo
2. ahol
3. alho
4. aloh
5. aohl
6. aolh
7. halo
8. haol
9. hlao
10. hloa
```

Ejercicio 02:

Cifrado por Filas:

```
Ingrese el mensaje a cifrar: CerditoFeliz
Ingrese la clave (número de filas n): 5

Matriz original:
C E R D I
T O F E L
I Z * * *
* * * * *
* * * * *

Matriz permutada:
* * * * *
C E R D I
T O F E L
I Z * * *
* * * * *

--- Resultados ---
Mensaje original: CERDITOFELIZ
Mensaje cifrado: *CTI**EOZ**RF***DE***IL**
Clave para descifrar (n): 5
```

Ejercicio 03:

Ejemplo en Python:

```python
mensaje = input("Ingrese el mensaje a cifrar: ").replace(" ", "").upper()
longitud = len(mensaje)
n = math.ceil(math.sqrt(longitud))

print(f"\nLongitud del mensaje: {longitud} caracteres")
print(f"Número de columnas: {n}")

mensaje_relleno = mensaje.ljust(n*n, '*')
```

Por su puesto primero tomamos el mensaje y eliminamos los espacios entre palabras, además tomamos cuantas columnas se va a usar, siempre dando suficiente espacio para el mensaje, por ejemplo, un n=3 nos dará una matriz 3x3 que tiene 9 espacio para las letras, entre más largo el mensaje se generarán más columnas, que será más seguro y por último rellenamos los espacios vacíos con *.

```python
matriz = []
for i in range(0, n*n, n):
    fila = list(mensaje_relleno[i:i+n])
    matriz.append(fila)
```

Creamos la matriz con el mensaje original y el n calculado y rellenamos los espacios.

```
# Mostrar matriz original
print("\nMatriz de cifrado:")
for fila in matriz:
    print(" ".join(fila))

# Generar permutación aleatoria de columnas
columnas_cifradas = random.sample(range(n), n)
print("\nclave:", columnas_cifradas)
```

Mostramos la matriz para cifrar y permutamos las columnas de forma aleatoria.

```
mensaje_cifrado = ""
for col in columnas_cifradas:
    for fila in matriz:
        mensaje_cifrado += fila[col]
```

Obtenemos el mensaje de la matriz permutada y lo ordenamos obteniendo el mensaje cifrado.

```
# Resultados
print("\nMensaje original:", mensaje)
print("Mensaje cifrado:", mensaje_cifrado)
```

Mostramos los dos mensajes para mostrar un antes y después.

```
Ingrese el mensaje a cifrar: hola mundo

Longitud del mensaje: 9 caracteres
Número de columnas: 3

Matriz de cifrado:
H O L
A M U
N D O

clave: [2, 0, 1]

Mensaje original: HOLAMUNDO
Mensaje cifrado: LUOHANOMD
```

Aquí un ejemplo con hola mundo, gracias a la permutación aleatoria el mismo mensaje tendrá diferente clave.

Ejercicio 04:

Ejemplo:

Alfabeto: ABCDEFGHIJKLMNOPQRSTUVWXYZ

Desplazamiento N: 3

Texto ingresado: manzana

```python
def main():
    # Entrada
    texto = input("Ingresa la cadena a cifrar: ")
    try:
        n = int(input("Ingresa el valor de desplazamiento n (entero): "))
    except ValueError:
        print("Desplazamiento inválido. Usa un número entero.")
        return

    # Generar alfabetos
    alfabeto_orig, alfabeto_cif = crear_alfabetos(n)

    # Cifrar
    texto_cifrado = cifrar_texto(texto, alfabeto_orig, alfabeto_cif)

    # Salida
    print(f"\nAlfabeto original: {alfabeto_orig}")
    print(f"Alfabeto cifrado : {alfabeto_cif}")
    print(f"Texto ingresado  : {texto}")
    print(f"Texto cifrado    : {texto_cifrado}")

if __name__ == "__main__":
    main()
```

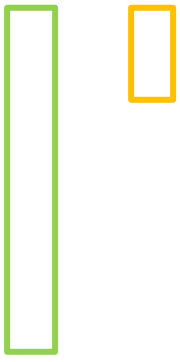
```
Ingresa la cadena a cifrar: manzana
Ingresa el valor de desplazamiento n (entero): 3

Alfabeto original: ABCDEFGHIJKLMNOPQRSTUVWXYZ
Alfabeto cifrado : XYZABCDEFGHIJKLMNOPQRSTUVW
Texto ingresado  : manzana
Texto cifrado    : jxkwxkx
```

Ejercicio 05:

EJEMPLO1:

```
Escribe tu mensaje: pagina
Palabra Clave: hola
```



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

```
El mensaje es: PAGINA
Clave_Vigenère: HOLAHO
Criptado_Vigenère: WORIUO
Desencriptado_Vigenère: PAGINA
```

Ejercicio 06:

Ejemplo:

Ejercicio 6: Algoritmo que realice el cifrado de una cadena de caracteres utilizando la siguiente tabla de cifrado:

*	A	S	D	F	G
Q	a	b	c	d	e
W	f	g	h	i	j
E	k	l	m	n	o
R	p	q	r	s	t
T	u	v	x	y	z

La cadena de caracteres se ingresa al iniciar el programa. Si algún caracter del texto no existe en la matriz, coloque "**". Imprima la matriz de cifrado, el mensaje original y el mensaje cifrado.

El algoritmo presente se encarga de definir una matriz de 5x5 que contiene las 25 letras del alfabeto en minúsculas, omitiendo la letra w.

Establece las coordenadas de cifrado utilizando etiquetas de filas y columnas (Q,W,E,R,T para las filas y A,S,D,F,G para las columnas).

```
filas = ['Q', 'W', 'E', 'R', 'T']
columnas = ['A', 'S', 'D', 'F', 'G']

matriz = [
    ['a', 'b', 'c', 'd', 'e'],
    ['f', 'g', 'h', 'i', 'j'],
    ['k', 'l', 'm', 'n', 'o'],
    ['p', 'q', 'r', 's', 't'],
    ['u', 'v', 'x', 'y', 'z']
]
```

Genera un diccionario en el que se asocia cada letra con su código de coordenadas (por ejemplo la letra “a” le corresponde “QA”).

Solicita un mensaje al usuario, sin espacio y solo con letras en minúscula.

```
cifrado_dict = {}
for i in range(len(filas)):
    for j in range(len(columnas)):
        letra = matriz[i][j]
        clave = filas[i] + columnas[j]
        cifrado_dict[letra] = clave

print("Matriz de Cifrado:")
for i in range(len(matriz)):
    for j in range(len(matriz[i])):
        print(f"{filas[i]}{columnas[j]}: {matriz[i][j]}", end="\t")
    print()

mensaje = input("\nIngresa el mensaje a cifrar (minúsculas sin espacios): ")
mensaje_cifrado = ""
```

Cifra el mensaje sustituyendo cada letra por su clave de coordenadas.

Si encuentra una letra no válida, la sustituye por “**” para indicar un carácter no reconocido.

Imprime la matriz de cifrado, el mensaje original y el mensaje cifrado.

```python
for c in mensaje:
    if c in cifrado_dict:
        mensaje_cifrado += cifrado_dict[c]
    else:
        mensaje_cifrado += "**"

print("\nMensaje original:", mensaje)
print("Mensaje cifrado:", mensaje_cifrado)
```

```
Matriz de Cifrado:
QA: a   QS: b   QD: c   QF: d   QG: e
WA: f   WS: g   WD: h   WF: i   WG: j
EA: k   ES: l   ED: m   EF: n   EG: o
RA: p   RS: q   RD: r   RF: s   RG: t
TA: u   TS: v   TD: x   TF: y   TG: z

Ingresa el mensaje a cifrar (minúsculas sin espacios): electroencefalografista

Mensaje original: electroencefalografista
Mensaje cifrado: QGESQGQDRGRDEGQGEFQDQGWAQAESEGWSRDQAWAWFRFRGQA
```