**INFORME PRELIMINAR DEL DESAFIO 1**

**JEAN CARLOS PARRA SERRANO**

**EDWIN SETYI DIAZ PILLIMUE**

**UNIVERSIDAD DE ANTIOQUIA**

**INFORMÁTICA 2**

**SEPTIEMBRE DEL 2025**

**MEDELLÍN, ANTIOQUIA**

***Contextualización***

El desafío plantea un problema de ingeniería inversa con archivos que fueron sometidos a dos procesos consecutivos: compresión y encriptación. El mensaje original en texto plano se comprimió usando RLE (Run-Length Encoding) o LZ78 (Lempel–Ziv 78). Luego, el resultado fue encriptado aplicando dos operaciones sobre cada byte:

1. Una rotación de bits hacia la izquierda en una cantidad fija n, con 0 < n < 8.

2. Una operación XOR con una clave K de un byte.

La tarea consiste en revertir este proceso para recuperar el mensaje original, lo cual implica:

- Identificar qué método de compresión se utilizó (RLE o LZ78).

- Determinar los parámetros de encriptación: la rotación de bits n y la clave XOR K.

- Desencriptar el archivo.

- Descomprimirlo para obtener el texto original completo.

Este desafío evalúa el dominio de programación en C++, incluyendo operaciones a nivel de bits, punteros, memoria dinámica y estructuras de control. Para hallar los valores de k y n, el cliente nos entrega un pequeño fragmento que está dentro del archivo encriptado.

***Análisis y diseño***

Para resolver el problema, definimos los pasos principales:

1. Lectura del archivo encriptado

- Se abre en modo binario para no perder información.

- Se almacena en un unsigned char\* mediante memoria dinámica.

2. Desencriptación tentativa

- Probar todas las combinaciones posibles de n (1 a 7) y K (0 a 255).

- Aplicar la operación inversa: primero XOR con K, luego rotación de bits hacia la derecha n.

3. Identificación del método de compresión

- Una vez desencriptado, el resultado debe corresponder a un formato válido de RLE o LZ78.

- RLE: se esperan secuencias de número + símbolo.

- LZ78: se esperan pares (índice, carácter) consistentes con un diccionario dinámico.

4. Uso de la pista

- Se cuenta con un fragmento del mensaje original en texto plano, provisto en archivos pistaX.

- Este fragmento sirve como verificación: al descomprimir, el mensaje reconstruido debe contener la pista.

- De esta manera, la pista nos permite validar qué combinación de parámetros (n, K) y qué algoritmo de compresión fueron realmente usados.

5. Descompresión

- Con los parámetros correctos ya identificados, se aplica la descompresión:

- Si es RLE, expandir repeticiones.

- Si es LZ78, reconstruir el texto usando el diccionario.

6. Resultado final

- Mostrar el mensaje original completo.

- Indicar el método de compresión, la clave K y el valor de n.

***Consideraciones adicionales***

- El texto original se limita a caracteres alfabéticos (A–Z, a–z) y dígitos (0–9), sin tildes, espacios ni símbolos especiales.

- El programa se debe implementar en C++ con Qt, respetando las restricciones:

- No usar std::string, STL ni struct.

- Uso de punteros, arreglos y memoria dinámica.

- El flujo de ejecución pedirá al usuario cuántos casos (n) se van a procesar, y por cada caso abrirá los archivos EncriptadoX.txt y pistaX.

***Nota:*** Se agrega que este es uno de los métodos propuestos primero. En los siguientes días, procesaremos si seguimos con esta versión de la solución o desarrollamos una solución a una propuesta nueva dada. Esta propuesta la planteamos de un orden inverso a la anterior. Consiste en recorrer la pista, comprimirla por los dos métodos, aplicarle los posibles valores de k y n y buscarla en el archivo que nos dieron. Esa es la idea planteada, veremos cómo se van desarrollando las etapas para determinar por cuál camino irnos.

**Conclusión**

Este proyecto combina conocimientos de algoritmos de compresión, criptografía básica y manejo de bajo nivel en C++. La estrategia general se basa en un proceso de prueba y validación, donde la pista es el elemento clave que permite confirmar los parámetros correctos y el algoritmo de compresión utilizado.