**Introducción**

Con el desafío I, se busca recrear un escenario de ingeniera inversa, en el que un mensaje ha sido sometido a un proceso de compresión y encriptación, donde la tarea consiste en reconstruirlo a partir de información parcial.

Para el desarrollo de esta actividad se debe contar con habilidades de programación en C++, desde el manejo de punteros y memoria dinámica, ya que estas serán útiles para la implementación de algoritmos de compresión (RLE Y LZ28). Además, de técnicas de técnicas de rotación de bits y operación XOR.

Con este trabajo se pretende poner en practica las habilidades adquiridas durante el proceso de formación, se pretende demostrar una metodología de solución coherente, eficiente y bien argumentada, que refleje las competencias adquiridas.

**Análisis y planteo de soluciones**

Teniendo en cuenta que tenemos un mensaje comprimido y encriptado, y sabemos que la compresión de este mensaje fue con RLE o LZ78, y luego se le aplico rotación de bits y XOR con una clave. Además, se cuenta con un fragmento conocido del mensaje original el cual nos servirá para identificar diferentes parámetros y determinar cuál es el método correcto para utilizar.

La dificultad para desarrollar este este desafío radica en que no conocemos ni el método de compresión empleado ni los parámetros de encriptación (valor de rotación y clave). Como se mencionó anteriormente lo único con lo que contamos es un fragmente conocido del mensaje en texto plano, que sirve simplemente para validar una posible solución,

En una primera instancia para desarrollar dicho desafío lo vamos a dividir en 4 partes principales o fases progresivas para una buena organización y desarrollo del código:

1. **Módulo de desencriptación**: El cual vamos a utilizar para revertir el mensaje encriptado ya sea con rotación de bits o XOR:
   * Explorar diferentes combinaciones de parámetros de rotación y clave.
   * Generar versiones candidatas del mensaje comprimido.
2. **Módulo de descompresión LZ78**: El cual se desarrolla mediante un diccionario dinámico el cual va constar de punteros dinámicos para reconstruir el mensaje a partir de pares (Índice, Carácter).
3. **Módulo de descompresión RLE**: Implemente una lógica la cual es reemplazar una secuencia de símbolos repetidos (un run).
4. **Módulo de integración o Main:** El cual será el encargado de unir todos estos métodos y nuestro flujo de datos, recibirá el mensaje encriptado, lo desencripta, prueba la descompresión con RLE y LZ78 y seleccionará la cual contenga el fragmento conocido.

Cuando se hagan pruebas de descompresión con las cadenas candidatas vamos a evaluar cual de ellas produce una salida coherente y legible.

Al momento de validar el fragmento conocido vamos a comparar el resultado de cada intento con el fragmento de el texto plano el cual nos va a servir de referencia. Además, identificar el conjunto de parámetros que produce dicho fragmento dentro del mensaje completo.

Por último, al momento de reconstruir nuestro mensaje final una vez encontrado el método y los parámetros correctos, vamos a aplicar todo el procedimiento inverso sobre el mensaje encriptado y así obtener un mensaje original.

**Consideraciones y conclusiones.**

Es importante tener en cuenta que no existe un único camino hacia la solución, se debe seguir un paso y proceso de prueba y error por la lógica y la validación del fragmento conocido.

La división en fases nos va a facilitar la implementación ya que podemos ir probando por fragmentos y nos ayudara a repartir responsabilidades dentro de nuestra pareja de trabajo.

Teniendo en cuenta que no se pueden utilizar librerías como STL nos lleva a hacer un análisis mas profundo entendiendo procesos como lo son manipulación de datos y memoria dinámica, reforzando el aprendizaje practico.

Es importante ser meticuloso al momento realizar la