**PROBLEMA C**

Se busca un algoritmo que permita saber si una línea de texto es permutación de otra.

**Algoritmo de solución:**

1. Se le asigna un valor dependiendo de la letra que se deba procesar a cada posición.
2. Se crea un ArrayList para cada conjunto de valores.
3. Dependiendo del valor que tenga cada posición se agrega en su ArrayList correspondiente.
4. Se agrega de forma ascendente los valores de cada ArrayList al ArrayList ordenado.
5. Llama el método de ordenamiento teniendo como parámetro las líneas de entrada.
6. Convierte cada ArrayList a un String.
7. Compara las dos cadenas de texto.

Para solucionar el problema se escogió como posible entrada únicamente las letras minúsculas de ‘a’ hasta la ‘z’ y el ‘ ‘; se convertía a una variable numérica que iba desde el 0 hasta 28 dependiendo a qué carácter fuera para que después se crearan 28 listas en donde se agregaban las diferentes posiciones del arreglo de números dependiendo de su valor, posteriormente si la lista no era vacía se agregaban sus valores a la lista ya ordenada, esto permite que se deba recorrer el arreglo solo una vez para ser convertido en una línea de texto con los valores ya ordenados.

**Análisis de complejidad:**

Hace referencia a cada una de las partes dividida en el código del archivo ProblemaB\_1.java

**Comentarios finales:**

Para mejorar la solución propuesta es necesario agregar los 228 caracteres que no fueron tomados en cuenta como variables de entrada, esto no tendría repercusión alguna en la complejidad del problema, ya que es el número de asignaciones las que se ven afectadas.

En cuanto a los caracteres Unicode según la solución propuesta sería aumentar el número de variables a comparar teniendo únicamente en cuenta las que no están en ASCII, pero si en Unicode. De esta manera la complejidad se mantendría ya que aumentarían las asignaciones que son , pero el tamaño del recorrido no es modificado porque sigue dependiendo del tamaño de la línea de texto.