1) Agregar a la clase **LinkedPositionalList.java** el siguiente método:

/\*\*

\* Remueve los elementos especificados en fromIndex inclusive (incluye la

\* posición dada) y toIndex exclusive (es uno menor a la posición dada). Si

\* fromIndex y toIndex son iguales, no remueve ningún elemento.

\*

\* **@param** fromIndex elemento inicial inclusive (incluye la posición dada)

\* **@param** toIndex elemento final exclusive (es uno menor a la posición dada)

\*

\* **@throws** IllegalArgumentException si fromIndex > toIndex

\* **@throws** IndexOutOfBoundsException si (fromIndex < 0 || toIndex > size)

\*/

/\*\*

\* Ejemplo:

\*

\* Dada la lista: [A, B, C, D, E]

\*

\* removeList(2,4)

\*

\* Queda la lista con los siguientes elementos: [A, B, E]

\*

\* Notar que elimina el elemento que está en la posición 2 (incluive) y no

\* elimina el elemento que está en la posición 4 (exclusive)

\*

\*/

**public** **void** removeAll(**int** fromIndex, **int** toIndex)

Realizar un programa que llame a ***removeAll*** con distintos parámetros probando las distintas condiciones que se pueden presentar y atrapar sus excepciones.

2) Agregar a la clase **LinkedPositionalList.java** el siguiente método:

/\*\*

\* Intercambia todos los elementos de la lista sin modificar sus posiciones

\*

\*/

/\*\*

\* Ejemplo:

\*

\* Dada la lista: [H, O, L, A]

\*

\* reverse()

\*

\* Se obtiene la lista: [A, L, O, H]

\*

\* Nota: no se modifican las posiciones. En este ejemplo, en la primer posición

\* está el elemento H después de llamar a reverse() en la misma posición esta el

\* elemento A

\*/

**void** reverse();

Realizar un programa que llame a ***reverse*** con distintas listas (lista con cantidad de elementos pares, cantidad de elementos impares, lista vacía, etc.) y mostrar los resultados. Se evaluará la eficiencia del algoritmo.