TRABAJO OBLIGATORIO DE ESTRUCTURAS DE DATOS Y ALGORITMOS

Se desea implementar un simulador de un editor de textos con capacidad de manejo de líneas y palabras, y con funcionalidades ligadas a un diccionario ortográfico de palabras. El sistema actuará a nivel de memoria no persistente y no contempla el desarrollo de interfaces gráficas.

Como características generales consideramos las siguientes:

1) El texto está compuesto por 0 o más líneas.
2) No hay límite en la cantidad de líneas a manejar.
3) La línea está compuesta por 0 o más palabras.
4) La cantidad máxima de palabras por línea está definida por una constante del sistema (MAX_CANT_PALABRAS_X_LINEA).
5) Las palabras no deben tener espacios en blanco.
6) En una línea, las palabras existentes deben ocupar posiciones consecutivas, desde la posición 1 (no hay huecos posibles entre palabras).
7) Se deben ignorar mayúsculas y minúsculas para las palabras. No obstante, los comandos del editor que muestren texto deben hacerlo según éste sea ingresado (respetando mayúsculas y minúsculas).
8) Al iniciar el sistema el texto será vacío (0 líneas) y el diccionario ortográfico no contendrá palabras.

Tipos de datos a manejar:

Cadena	`typedef char* Cadena;`

TipoRetorno	`enum _retorno{`
	`  OK, ERROR, NO_IMPLEMENTADA`
	`};`
	
	`typedef enum _retorno TipoRetorno;`

Posicion	`typedef unsigned int Posicion;`

Pueden definirse tipos de datos auxiliares.

El sistema debe permitir realizar las operaciones que detallamos a continuación. Definimos primero las operaciones relativas al texto (las líneas), luego las operaciones sobre una línea y finalmente las funcionalidades relativas al diccionario ortográfico de palabras. En cada caso describimos una funcionalidad del sistema, desarrollamos un ejemplo de la operación partiendo de un texto vacío, e indicamos los posibles retornos.

OPERACIONES RELATIVAS A LAS LÍNEAS (AL DOCUMENTO):

1) Inserta una nueva línea vacía al final del texto. Este requerimiento debe ser resuelto en O(1) peor caso.

```
TipoRetorno InsertarLinea();
```

Esta operación siempre debe insertar una nueva línea ya que no hay errores (lógicos) que puedan producirse.

```
Ejemplo:
      InsertarLinea();
      InsertarLinea();
      ImprimirTexto();

      Salida:
            1:
            2:
```

Nota: `ImprimirTexto` se desarrolla como operación 6 del sistema.

Retornos posibles:	
`OK`	• Se insertó la línea.
`NO_IMPLEMENTADA`	• Cuando aún no se implementó. Es el tipo de retorno por defecto.

2) Inserta una nueva línea vacía en la posición indicada.

```
TipoRetorno InsertarLineaEnPosicion(Posicion posicionLinea);
```

Inserta una línea vacía en la posición indicada y mueve todas las líneas que se encuentran a partir de la posición indicada, una posición más adelante.
La posición es válida solamente si (*posicionLinea* >= 1) y (*posicionLinea* <= cantidad de líneas + 1)

```
Ejemplo:
      InsertarLineaEnPosicion(1);
      InsertarPalabra(1, 1, "Palabra1");
      ImprimirTexto();

      Salida:
            1: Palabra1

      InsertarLineaEnPosicion(1);
      ImprimirTexto();

      Salida:
            1:
            2: Palabra1
```

Nota: `InsertarPalabra` se desarrolla como operación 8 del sistema.

Retornos posibles:	
OK	• Si se pudo insertar la línea con éxito.
ERROR	• Si la posición no es válida.
NO_IMPLEMENTADA	• Cuando aún no se implementó. Es el tipo de retorno por defecto.

3) Borra la línea en la posición indicada.

`TipoRetorno BorrarLinea(Posicion posicionLinea);`

Borra la línea en la posición indicada y mueve todas las líneas que se encuentran a partir de la posición indicada, una posición más hacia arriba.
La posición es válida solamente si posicionLinea existe en el texto, esto es, si *posicionLinea* >= 1 y *posicionLinea* <= cantidad de líneas.

```
Ejemplo:
      InsertarLinea();
      InsertarLinea();
      InsertarPalabra(2, 1, "Palabra1");
      ImprimirTexto();

      Salida:
            1:
            2: Palabra1


      BorrarLinea(1);
      ImprimirTexto();

      Salida:
            1: Palabra1
```

Retornos posibles:	
OK	• Si se pudo borrar la línea con éxito.
ERROR	• Si la posición no es válida.
NO_IMPLEMENTADA	• Cuando aún no se implementó. Es el tipo de retorno por defecto.

4) Borra todas las líneas del texto.

`TipoRetorno BorrarTodo();`

Borra todas las líneas del texto. No debe ocurrir ningún error al ejecutar la operación (si el texto era vacío, queda igual).

```
Ejemplo:
      InsertarLinea();
      InsertarLinea();
      InsertarPalabra(2, 1, "Palabra1");
      ImprimirTexto();

      Salida:
            1:
            2: Palabra1
```

```
BorrarTodo();
ImprimirTexto();

Salida:
      Texto vacio
```

Retornos posibles:	
OK	• Se borró el texto.
NO_IMPLEMENTADA	• Cuando aún no se implementó. Es el tipo de retorno por defecto.

5) Borra todas las ocurrencias de una palabra en el texto.

`TipoRetorno BorrarOcurrenciasPalabraEnTexto(Cadena palabraABorrar);`

Borra todas las ocurrencias en el texto de la palabra indicada, desplazando hacia adelante en cada línea las palabras que eventualmente se encuentren en posiciones posteriores a la eliminada.

```
Ejemplo:
      InsertarLinea();
      InsertarLinea();
      InsertarPalabra(1, 1, "Palabra1");
      InsertarPalabra(1, 2, "Palabra2");
      InsertarPalabra(2, 1, "Palabra1");
      InsertarPalabra(2, 1, "Palabra2");
      InsertarLineaEnPosicion(2);
      InsertarPalabra(2, 1, "Palabra2");
      ImprimirTexto();

      Salida:
            1: Palabra1 Palabra2
            2: Palabra2
            3: Palabra2 Palabra1


      BorrarOcurrenciasPalabraEnTexto("Palabra2");
      ImprimirTexto();

      Salida:
            1: Palabra1
            2:
            3: Palabra1
```

Retornos posibles:	
OK	• Si se pudieron borrar las ocurrencias de la palabra con éxito, incluso si no fue borrada ninguna palabra (por no pertenecer al texto).
NO_IMPLEMENTADA	• Cuando aún no se implementó. Es el tipo de retorno por defecto.

6) Imprime el texto por pantalla.

`TipoRetorno ImprimirTexto();`

Muestra todas las líneas del texto con sus respectivas palabras. Se debe imprimir el número de línea, para cada línea, según muestra el ejemplo. Cuando el texto no tiene líneas se debe mostrar el mensaje "Texto vacío".

```
Ejemplo:
      InsertarLinea();
      InsertarPalabra(1, 1, "Palabra1");
      InsertarLinea();
      InsertarPalabra(2, 1, "Palabra2");
      InsertarLinea();
      ImprimirTexto();

      Salida:
            1: Palabra1
            2: Palabra2
            3:

      BorrarTodo();
      ImprimirTexto();

      Salida:
            Texto vacío
```

Retornos posibles:	
OK	• Se mostró el texto.
NO_IMPLEMENTADA	• Cuando aún no se implementó. Es el tipo de retorno por defecto.

7) Comprime las palabras del texto. Para implementar esta operación no debe generarse un nuevo documento.

`TipoRetorno ComprimirTexto();`

Reubica las palabras en el texto de tal manera que, manteniendo la relación posicional entre palabras, todas las líneas del texto (excepto posiblemente la última) tengan su capacidad máxima de la palabras colmada. Naturalmente, el texto puede reducir su número de líneas luego de esta operación. No deberían quedar líneas vacías luego de comprimir un texto.

```
Ejemplo (suponiendo MAX_CANT_PALABRAS_X_LINEA igual a 3):
      InsertarLinea();
      InsertarLinea();
      InsertarLinea();
      InsertarLinea();
      InsertarLinea();
      InsertarPalabra(1, 1, "Palabra1");
      InsertarPalabra(1, 2, "Palabra2");
      InsertarPalabra(3, 1, "Palabra3");
      InsertarPalabra(4, 1, "Palabra4");
      InsertarPalabra(5, 1, "Palabra5");

      InsertarPalabra(5, 2, "Palabra6");
```

```
InsertarPalabra(5, 3, "Palabra7");


ImprimirTexto();

Salida:
      1: Palabra1 Palabra2
      2:
      3: Palabra3
      4: Palabra4
      5: Palabra5 Palabra6 Palabra7


ComprimirTexto();
ImprimirTexto();

Salida:
      1: Palabra1 Palabra2 Palabra3
      2: Palabra4 Palabra5 Palabra6
      3: Palabra7
```

Retornos posibles:	
OK	• Si se compactó el texto.
NO_IMPLEMENTADA	• Cuando aún no se implementó. Es el tipo de retorno por defecto.

OPERACIONES RELATIVAS A LAS PALABRAS:

8) Inserta una palabra en una línea.

`TipoRetorno InsertarPalabra(Posicion posicionLinea, Posicion posicionPalabra, Cadena palabraAIngresar);`

Inserta la palabra *palabraAIngresar* en la línea *posicionLinea* y dentro de la línea en la posición *posicionPalabra*. Desplaza todas las palabras que se encuentran a partir de la posición *posicionPalabra* un lugar hacia adelante. Si al ingresar la palabra se superara la cantidad máxima de palabras por línea (MAX_CANT_PALABRAS_X_LINEA), el desplazamiento afectaría a la línea siguiente y eventualmente a las posteriores, llegando incluso a ser necesario en el caso extermo agregar una nueva línea al final del documento con una sóla palabra (si todas las líneas posteriores estuvieran llenas).

La posición *posicionLinea* es válida solamente si existe en el texto.
La posición *posicionPalabra* es válida solamente si (*posicionPalabra* >= 1) y (*posicionPalabra* <= cantidad de palabras en la línea + 1).

```
Ejemplo (suponiendo MAX_CANT_PALABRAS_X_LINEA igual a 3):
      InsertarLinea();
      InsertarLinea();
      InsertarPalabra(2, 1, "Palabra1");
      InsertarPalabra(2, 2, "Palabra2");
      ImprimirTexto();



      Salida:
```

```
            1:
            2: Palabra1 Palabra2

      InsertarPalabra(2, 1, "Palabra3");
      ImprimirTexto();

      Salida:
            1:
            2: Palabra3 Palabra1 Palabra2

      InsertarPalabra(2, 2, "Palabra4");
      ImprimirTexto();

      Salida:
            1:
            2: Palabra3 Palabra4 Palabra1
            3: Palabra2

      InsertarPalabra(2, 3, "Palabra5");
      ImprimirTexto();

      Salida:
            1:
            2: Palabra3 Palabra4 Palabra5
            3: Palabra1 Palabra2
```

Retornos posibles:	
OK	• Si se pudo insertar la palabra con éxito.
ERROR	• Si la posición de la línea no es válida. • Si la posición de la palabra no es válida.
NO_IMPLEMENTADA	• Cuando aún no se implementó. Es el tipo de retorno por defecto.

9) Borra la palabra en la posición indicada.

`TipoRetorno BorrarPalabra(Posicion posicionLinea, Posicion posicionPalabra);`

Borra la palabra en la posición *posicionPalabra* de la línea *posicionLinea* y mueve todas las palabras (si las hay) en las posiciones posteriores de dicha línea un lugar hacia adelante.
La *posicionLinea* es válida solamente si *posicionLinea* existe en el texto.
La *posicionPalabra* es válida solamente si *posicionPalabra* existe en la línea.

```
Ejemplo:
      InsertarLinea();
      InsertarLinea();
      InsertarPalabra(2, 1, "Palabra1");
      InsertarPalabra(2, 2, "Palabra2");
      InsertarPalabra(2, 1, "Palabra3");
      ImprimirTexto();

      Salida:
            1:
            2: Palabra3 Palabra1 Palabra2

      BorrarPalabra(2, 1);
      ImprimirTexto();
```

```
Salida:
      1:
      2: Palabra1 Palabra2

InsertarLinea();
InsertarPalabra(3, 1, "Palabra3");
BorrarPalabra(2, 1);
BorrarPalabra(2, 1);
ImprimirTexto();

Salida:
      1:
      2:
      3: Palabra3
```

Retornos posibles:	
OK	• Si se pudo borrar la palabra con éxito.
ERROR	• Si la posición de la línea no es válida. • Si la posición de la palabra no es válida.
NO_IMPLEMENTADA	• Cuando aún no se implementó. Es el tipo de retorno por defecto.

10) Borra todas las ocurrencias de una palabra en la línea indicada.

`TipoRetorno BorrarOcurrenciasPalabraEnLinea(Posicion posicionLinea, Cadena palabraABorrar);`

Borra todas las ocurrencias de la palabra *palabraABorrar* en la línea indicada. Cada vez que borra una palabra mueve todas las palabras de la línea que se encuentran a partir de la posición borrada una posición hacia delante.
La *posicionLinea* es válida solamente si *posicionLinea* existe en el texto.

```
Ejemplo:
      InsertarLinea();
      InsertarLinea();
      InsertarPalabra(2, 1, "Palabra1");
      InsertarPalabra(2, 2, "Palabra2");
      InsertarPalabra(2, 1, "Palabra2");
      ImprimirTexto();

      Salida:
            1:
            2: Palabra2 Palabra1 Palabra2

      BorrarOcurrenciasPalabraEnLinea(2, "Palabra2");
      ImprimirTexto();

      Salida:
            1:
            2: Palabra1
```

Retornos posibles:	
OK	• Si se pudieron borrar las ocurrencias de la palabra con éxito, incluso si no

	fue borrada ninguna palabra (por no pertenecer a la línea indicada).
ERROR	• Si la posición de la línea no es válida.
NO_IMPLEMENTADA	• Cuando aún no se implementó. Es el tipo de retorno por defecto.

11) Imprime la línea por pantalla.

```
TipoRetorno ImprimirLinea(Posicion posicionLinea);
```

Imprime la línea con todas sus palabras. Se debe imprimir el número de línea según muestra el ejemplo. La *posicionLinea* es válida solamente si *posicionLinea* existe en el texto.

```
Ejemplo:
      InsertarLinea();
      InsertarLinea();
      ImprimirLinea(2);

      Salida:
            2:

      InsertarPalabra(2, 1, "Palabra1");
      InsertarPalabra(2, 2, "Palabra2");
      ImprimirLinea(2);

      Salida:
            2: Palabra1 Palabra2
```

Retornos posibles:	
OK	• Si se mostró la línea.
ERROR	• Si la posición de la línea no es válida.
NO_IMPLEMENTADA	• Cuando aún no se implementó. Es el tipo de retorno por defecto.

OPERACIONES RELATIVAS AL DICCIONARIO:

12) Agrega una palabra al diccionario. Esta operación debe realizarse en a lo sumo $O(\log_2 n)$ promedio.

```
TipoRetorno IngresarPalabraDiccionario(Cadena palabraAIngresar);
```

Ingresa una palabra al diccionario si ésta no se encuentra en el mismo.

```
Ejemplo:
      IngresarPalabraDiccionario("Hoja");
      IngresarPalabraDiccionario("Hojalata");
      IngresarPalabraDiccionario("Bosque");
      ImprimirDiccionario();

      Salida:
            Bosque
            Hoja
            Hojalata
```

Nota: `ImprimirDiccionario` se desarrolla como operación 14 del sistema.

Retornos posibles:	
OK	• Si se ingresó correctamente.
ERROR	• Si la palabra ya existe.

NO_IMPLEMENTADA	• Cuando aún no se implementó. Es el tipo de retorno por defecto.

13) Borra una palabra del diccionario.

`TipoRetorno BorrarPalabraDiccionario(Cadena palabraABorrar);`

Borra una palabra del diccionario si se encuentra en el mismo.

```
Ejemplo:
      IngresarPalabraDiccionario("Hoja");
      IngresarPalabraDiccionario("Hojalata");
      IngresarPalabraDiccionario("Bosque");
      BorrarPalabraDiccionario("Hoja");
      ImprimirDiccionario();

      Salida:
            Bosque
            Hojalata
```

Retornos posibles:	
OK	• Si se borró correctamente.
ERROR	• Si la palabra no existe.
NO_IMPLEMENTADA	• Cuando aún no se implementó. Es el tipo de retorno por defecto.

14) Muestra las palabras del diccionario alfabéticamente. Esta operación debe realizarse en O(n) peor caso.

`TipoRetorno ImprimirDiccionario();`

Muestra las palabras del diccionario ordenadas alfabéticamente, de menor a mayor. Debe imprimirse según muestra el ejemplo. Cuando el diccionario no tiene palabras debe mostrarse el mensaje "Diccionario vacío".

```
Ejemplo:
      ImprimirDiccionario();

      Salida:
            Diccionario vacio

      IngresarPalabraDiccionario("Hoja");
      IngresarPalabraDiccionario("Hojalata");
      IngresarPalabraDiccionario("Bosque");
      ImprimirDiccionario();

      Salida:
            Bosque
            Hoja
            Hojalata
```

Retornos posibles:	
OK	• Si se mostró el diccionario.
NO_IMPLEMENTADA	• Cuando aún no se implementó. Es el tipo de retorno por defecto.

15) Muestra las palabras del texto que no se encuentran en el diccionario.

`TipoRetorno ImprimirTextoIncorrecto();`

Muestra todas las líneas del texto, pero exhibiendo solamente las palabras que no se encuentran en el dicionario. Se debe imprimir el número de línea según muestra el ejemplo. Cuando el texto no tiene líneas se debe mostrar el mensaje "Texto vacio".

```
Ejemplo:
      InsertarLinea();
      InsertarLinea();
      InsertarLinea();
      InsertarPalabra(2, 1, "Palabra21");
      InsertarPalabra(2, 2, "Palabra22");
      InsertarPalabra(1, 1, "Palabra11");
      InsertarPalabra(1, 2, "Palabra12");
      InsertarPalabra(1, 3, "Palabra13");
      InsertarPalabra(3, 1, "Palabra31");
      ImprimirTexto();

      Salida:
            1: Palabra11 Palabra12 Palabra13
            2: Palabra21 Palabra22
            3: Palabra31


      IngresarPalabraDiccionario("Palabra12");
      IngresarPalabraDiccionario("Palabra21");
      IngresarPalabraDiccionario("Palabra22");
      ImprimirTextoIncorrecto();

      Salida:
            1: Palabra11 Palabra13
            2:
            3: Palabra31
```

Retornos posibles:	
OK	• Se mostró el texto.
NO_IMPLEMENTADA	• Cuando aún no se implementó. Es el tipo de retorno por defecto.

16) Imprime las últimas palabras ingresadas. Esta operación debe realizarse eficientemente, sin implicar además una sobrecarga, más que en un valor constante, sobre el tiempo de ejecución de la operación 8.

`TipoRetorno ImprimirUltimasPalabras();`

Imprime las últimas MAX_CANT_ULTIMAS_PALABRAS, a lo sumo, palabras ingresadas al texto (no al diccionario), aún cuando estas palabras pudieran haber sido luego eliminadas.
MAX_CANT_ULTIMAS_PALABRAS es una constante del sistema.
El orden en el que se muestran las palabras debe ser el siguiente: primero la última palabra ingresada, luego la penúltima, y así sucesivamente. Si nunca se ingresaron palabras se debe mostrar "No se ingresaron palabras".

```
Ejemplo (suponiendo MAX_CANT_ULTIMAS_PALABRAS igual a 3):
      InsertarLinea();
      InsertarLinea();
      ImprimirUltimasPalabras();
```

```
Salida:
      No se ingresaron palabras

InsertarPalabra(2, 1, "Palabra21");
InsertarPalabra(2, 2, "Palabra22");
IngresarPalabraDiccionario("Hoja");
ImprimirTexto();
Salida:
      1:
      2: Palabra21 Palabra22

ImprimirUltimasPalabras();

Salida:
      Palabra22
      Palabra21

InsertarPalabra(1, 1, "Palabra11");
InsertarPalabra(1, 2, "Palabra12");
ImprimirTexto();

Salida:
      1: Palabra11 Palabra12
      2: Palabra21 Palabra22

ImprimirUltimasPalabras();

Salida:
      Palabra12
      Palabra11
      Palabra22
```

Retornos posibles:	
OK	• Se mostraron las, a lo sumo, últimas MAX_CANT_ULTIMAS_PALABRAS palabras ingresadas al texto.
NO_IMPLEMENTADA	• Cuando aún no se implementó. Es el tipo de retorno por defecto.

CATEGORÍA DE OPERACIONES

TIPO 1	Operaciones **imprescindibles** para que el trabajo obligatorio sea corregido.
TIPO 2	Operaciones importantes. Estas serán probadas independientemente, siempre que estén correctamente implementadas las operaciones de TIPO 1.
OPCIONAL	Operación, que realizada correctamente, acumula hasta 10 puntos adicionales.

Tipo 1

- **1 InsertarLinea**
- **2 InsertarLineaEnPosicion**
- **5 BorrarOcurrenciasPalabraEnTexto**
- **6 ImprimirTexto**
- **8 InsertarPalabra**
- **9 BorrarPalabra**
- **10 BorrarOcurrenciasPalabraEnLinea**
- **11 ImprimirLinea**
- **12 IngresarPalabraDiccionario**
- **14 ImprimirDiccionario**

Tipo 2

- **3 BorrarLinea**
- **4 BorrarTodo**
- **13 BorrarPalabraDiccionario**
- **15 ImprimirTextoIncorrecto**

Opcional

- **7 ComprimirTexto**
- **16 ImprimirUltimasPalabras**

SOBRE LAS ENTREGAS

- El obligatorio se debe hacer en grupos de 3 alumnos.
- La aprobación del obligatorio es individual, no grupal.
- Durante el semestre se coordinaran instancias con los grupos para ver el avance del trabajo y evacuar las dudas que puedan surgir.
- Se deben entregar los archivos .c y .h de la solución y un makefile que compile el proyecto.

A fin de construir el simulador del editor de textos se pide:

Desarrollo del sistema siguiendo la metodología vista en el curso.

1. ***Primera parte:*** operaciones 1-11 – ***Seria ideal tenerla lista para el 26 de octubre del 2021 (coordinar monitoreo con el docente)***
2. ***Segunda parte:*** operaciones 12-16 - ***Entrega Final 20 de noviembre del 2021 (coordinar defensa con el docente)***