UNIVERSIDAD ORT
Uruguay

Facultad de Ingeniería
Bernard Wand-Polak
Cuareim 1451
11.100 Montevideo, Uruguay
Tel. 2902 15 05 Fax 2908 13 70
www.ort.edu.uy

EVALUACION	Obligatorio	GRUPO	TODOS	**FECHA**	Mayo de 2022
MATERIA	**Algoritmos y Estructuras de Datos 1**				
CARRERA	Analista Programador / Analista en Tecnologías de la información				
CONDICIONES	**- Puntos:** Máximo: 40 Mínimo: 0 **- Fecha máxima de entrega**: **16/06/2022 hasta las 21 hs.** LA ENTREGA SE REALIZA EN FORMA ONLINE EN ARCHIVO NO MAYOR A 40MB EN FORMATO ZIP O PDF. **IMPORTANTE** - Los grupos deben estar conformados por hasta 3 personas. - Inscribirse - Subir el trabajo a Gestión antes de la hora indicada, ver hoja al final del documento: "RECORDATORIO" - Se debe entregar un conjunto de pruebas (en JUnit o utilizando la clase Prueba y Retorno provista durante el curso). Estas deben cubrir todos los métodos, tanto los casos con resultado OK, como los de ERROR.				

UNIVERSIDAD ORT
Uruguay

Facultad de Ingeniería
Bernard Wand-Polak
Cuareim 1451
11.100 Montevideo, Uruguay
Tel. 2902 15 05 Fax 2908 13 70
www.ort.edu.uy

Obligatorio: Telegramas

Contenido

Facultad de Ingeniería
Bernard Wand-Polak
Cuareim 1451
11.100 Montevideo, Uruguay
Tel. 2902 15 05 Fax 2908 13 70
www.ort.edu.uy

1. Introducción

Se desea implementar un prototipo que simule un editor de textos para una app de mensajería instantánea.

La app deberá soportar la gestión de contactos (alta, baja y listado) y el manejo de mensajes que éstos envían y reciben (agregar líneas, palabras y su vinculación a un diccionario ortográfico de palabras).

Cada mensaje debe estar vinculado al contacto/destinatario de manera de poder ver un contacto y todos los mensajes/diálogos mantenidos con el mismo.

A modo de ejemplo cada contacto contendrá un conjunto de mensajes y cada mensaje estará compuesto por un conjunto de líneas que a su vez se compone de un conjunto de palabras respetando las pautas que se describirán a continuación.

Cada mensaje de texto puede estar compuesto por más de una línea no habiendo limite en cuanto a la cantidad de líneas a manejar. Cada línea está compuesta por palabras con un límite máximo de palabras (MAX_CANT_PALABRAS_X_LINEA) que estará definido por el sistema.

Algunas consideraciones:

- Al iniciar el sistema, el diccionario ortográfico no contendrá palabras.
- Al crear un mensaje, el texto será vacío (0 líneas).
- Las palabras no deben tener espacios en blanco.
- En una línea, las palabras existentes deben ocupar posiciones consecutivas, comenzando desde la posición 1 en adelante (no hay huecos posibles entre palabras).

El sistema actuará a nivel de memoria no persistente y no contempla el desarrollode interfaces gráficas.

Se proveen los siguientes tipos de datos que deberán ser respetados.

Sistema	public class Sistema{
	/*Aquí introduzca la información que estime conveniente*/
	}

Retorno	public class Retorno{
	enum Retorno{OK,ERROR, NO_IMPLEMENTADA};
	/*Aquí introduzca la información que estime conveniente*/
	
	boolean valorBooleano
	int valorEntero;
	String valorString;
	Resultado resultado;
	
	}

Pueden definirse tipos de datos (clases) auxiliares.

2. Funcionalidades principales

2.1. Crear Sistema de Mensajes

Firma: `Retorno crearSistemaMensajes(int MAX_CANT_PALABRAS_X_LINEA);`

Descripción: Crea la estructura necesaria para representar el sistema de mensajes.

Retornos posibles	
OK	Siempre retorna OK.
ERROR	No hay errores
NO_IMPLEMENTADA	Cuando aún no se implementó. Es el tipo de retorno por defecto.

Nota: el sistema debe recibir como parámetro la cantidad máxima de palabras por línea que acepta el mensaje, donde: MAX_CANT_PALABRAS_X_LINEA >= 1.

2.2. Destruir Sistema de Mensajes

Firma: `Retorno destruirSistemaMensajes();`

Descripción: Destruye el sistema de mensajes y todos sus elementos, liberando la memoria utilizada.

Retornos posibles	
OK	Siempre retorna OK.
ERROR	No existen errores posibles.
NO_IMPLEMENTADA	Cuando aún no se implementó. Es el tipo de retorno por defecto.

2.3. Crear Contacto

Firma: `Retorno agregarContacto(int numContacto, String nomContacto);`

Descripción: Agrega el contacto a la lista de contactos, siempre y cuando ya no exista un contacto con dicho número de contacto.

Retornos posibles	
OK	Si pudo agregar el contacto a la lista de contactos.
ERROR	Si el contacto ya existe.
NO_IMPLEMENTADA	Cuando aún no se implementó. Es el tipo de retorno por defecto.

2.4. Eliminar Contacto

Firma: `Retorno eliminarContacto(int numContacto);`

Descripción: Elimina el contacto con el número de contacto indicado.

Retornos posibles	
OK	Si pudo eliminar el contacto de la lista de contactos.
ERROR	En caso de que no exista un contacto con el número indicado.
NO_IMPLEMENTADA	Cuando aún no se implementó. Es el tipo de retorno por defecto.

2.5. Agregar Mensaje

Firma: `Retorno agregarMensaje(int numContactoOrigen, int numContactoDestino,Date fecha);`

Descripción: Crear y vincula el mensaje con los contactos correspondientes.

Retornos posibles	
OK	Si pudo crear y vincular el mensaje.
ERROR	En caso de que alguno de los contactos no exista.
NO_IMPLEMENTADA	Cuando aún no se implementó. Es el tipo de retorno por defecto.

Nota: los mensajes serán numerados en forma automática para cada contacto origen en forma correlativa comenzando del número 1.

2.6. Eliminar Mensaje

Firma: `Retorno eliminarMensaje(int numContactoOrigen, int numMensaje);`

Descripción: Elimina el mensaje vinculado a los contactos involucrados.

Retornos posibles	
OK	Si pudo liminar el contacto.
ERROR	En caso de que no exista el número de contacto o que no exista el número de mensaje para el contacto indicado.
NO_IMPLEMENTADA	Cuando aún no se implementó. Es el tipo de retorno por defecto.

3. Operaciones relativas a los mensajes

3.1. Imprime el texto por pantalla.

Firma: `Retorno imprimirTexto(int numContactoOrigen, int numMensaje);`

Descripción: Muestra todas las líneas del mensaje 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".

Retornos posibles	
OK	Se muestra texto con éxito.
ERROR	En caso de que no exista el número de contacto o que no exista el número de mensaje para el contacto indicado.
NO_IMPLEMENTADA	Cuando aún no se implementó. Es el tipo de retorno por defecto.

Ejemplo para un mensaje con 3 líneas (las dos primeras con 2 palabras cada una y la tercera sin palabras):

Salida:

1: Palabra1 Palabra2
2: Palabra3 Palabra4
3:

Ejemplo para texto si líneas definidas:

Salida:

Texto vacio

3.2. Inserta una nueva línea vacía al final del texto del mensaje.

Firma: `Retorno insertarLinea(int numContactoOrigen, int numMensaje);`

Descripción: Inserta una nueva línea vacía al final del texto.

Retornos posibles	
OK	Esta operación siempre debe insertar una nueva línea
ERROR	En caso de que no exista el número de contacto o que no exista el número de mensaje para el contacto indicado.
NO_IMPLEMENTADA	Cuando aún no se implementó. Es el tipo de retorno por defecto.

Ejemplo:

```
InsertarLinea();
InsertarLinea();
ImprimirTexto()
```

Salida:

```
1:
2:
```

Nota: Esta operación debe ser realizada en forma recursiva

3.3. Inserta una nueva línea vacía en la posición indicada.

Firma: Retorno insertarLineaEnPosicion(int numContactoOrigen, int numMensaje, int posicionLinea);

Descripción: 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 actuales + 1).

Retornos posibles	
OK	Si se pudo insertar la línea con éxito.
ERROR	En caso de que no exista el número de contacto o que no exista el número de mensaje para el contacto indicado. Si la posición no es valida
NO_IMPLEMENTADA	Cuando aún no se implementó. Es el tipo de retorno por defecto.

Ejemplo:

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

Salida:

```
1: Palabra1
InsertarLineaEnPosicion(1);
ImprimirTexto();
```

Facultad de Ingeniería
Bernard Wand-Polak
Cuareim 1451
11.100 Montevideo, Uruguay
Tel. 2902 15 05 Fax 2908 13 70
www.ort.edu.uy

Salida:

1:
2: Palabra1

Nota: InsertarPalabra se desarrolla como operación 8 del sistema

3.4. Borra la línea en la posición indicada.

Firma: `Retorno borrarLinea(int numContactoOrigen, int numMensaje, int posicionLinea);`

Descripción: 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).

Retornos posibles	
OK	Si se pudo borrar la línea con éxito.
ERROR	En caso de que no exista el número de contacto o que no exista el número de mensaje para el contacto indicado. Si la posición no es valida
NO_IMPLEMENTADA	Cuando aún no se implementó. Es el tipo de retorno por defecto.

Ejemplo:

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

Salida:
1:
2: Palabra1

BorrarLinea(1);
ImprimirTexto();

Salida:
1: Palabra1
```

3.5. Borra todas las líneas del texto.

Firma: `Retorno borrarTodo(int numContactoOrigen, int numMensaje);`

Descripción: Borra todas las líneas del texto.

Retornos posibles	
OK	Si se pudo borrar el texto con éxito.
ERROR	En caso de que no exista el número de contacto o que no exista el número de mensaje para el contacto indicado. Si la posición no es valida
NO_IMPLEMENTADA	Cuando aún no se implementó. Es el tipo de retorno por defecto.

Ejemplo:

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

Salida:

```
1:
2: Palabra1
```

```
borrarTodo();
ImprimirTexto();
```

Salida:

```
Texto vacio
```

Nota: Esta operación debe ser realizada en forma recursiva

3.6. Borra todas las ocurrencias de una palabra en el texto.

Firma: `Retorno borrarOcurrenciasPalabraEnTexto(int numContactoOrigen, int numMensaje, String palabraABorrar);`

Descripción: 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.

Retornos posibles	
OK	Si se pudieron borrar las ocurrencias de la palabra con éxito.
ERROR	En caso de que no exista el número de contacto o que no exista el número de mensaje para el contacto indicado. Si la palabra a borrar no existe
NO_IMPLEMENTADA	Cuando aún no se implementó. Es el tipo de retorno por defecto.

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
```

Nota: Esta operación debe ser realizada en forma recursiva

3.7. Inserta una palabra en una línea.

Firma: `Retorno insertarPalabraEnLinea(int numContactoOrigen, int numMensaje, int posicionLinea, int posicionPalabra, String palabraAIngresar);`

Descripción: Inserta la palabra palabraAIngresar en la línea posicionLinea y dentro de la línea en la posición posicionPalabra. Si al ingresar la palabra se superara la cantidad máxima de palabras por línea (MAX_CANT_PALABRAS_X_LINEA), la inserción no ser haría y devolvería un mensaje de error. 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).

Retornos posibles	
OK	Se pudo insertar palabra con éxito.
ERROR	En caso de que no exista el número de contacto o que no exista el número de mensaje para el contacto indicado. Si la posición de la línea no es válida. Si la posición de la palabra no es válida. Si se superara la cantidad máxima de palabras por línea.
NO_IMPLEMENTADA	Cuando aún no se implementó. Es el tipo de retorno por defecto.

Ejemplo (suponiendo MAX_CANT_PALABRAS_X_LINEA igual a 3):

```
InsertarLinea();
InsertarLinea();
InsertarPalabraEnLinea (2, 1, "Palabra1");
InsertarPalabraEnLinea (2, 2, "Palabra2");
ImprimirTexto();
```

Salida:

```
1:
2: Palabra1 Palabra2

InsertarPalabraEnLinea (2, 1, "Palabra3");
ImprimirTexto();
```

Salida:

```
1:
2: Palabra3 Palabra1 Palabra2

InsertarPalabraYDesplazar(2, 2, "Palabra4");
ImprimirTexto();
```

Salida:

```
1:
2: Palabra3 Palabra4 Palabra1
3: Palabra2
InsertarPalabraYDesplazar (2, 3, "Palabra5");
ImprimirTexto();
```

Salida:

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

3.8. Inserta una palabra en una línea y desplaza a la siguiente si es necesario.

Firma: `Retorno insertarPalabraYDesplazar(int numContactoOrigen, int numMensaje, int posicionLinea, int posicionPalabra, String palabraAIngresar);`

Descripción: 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 extremo agregar una nueva línea al final del documento con una sola 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).

Retornos posibles	
OK	Se pudo insertar palabra con éxito.
ERROR	En caso de que no exista el número de contacto o que no exista el número de mensaje para el contacto indicado. Si la posición de la línea no es válida. 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.

Ejemplo (suponiendo MAX_CANT_PALABRAS_X_LINEA igual a 3):

```
InsertarLinea();
InsertarLinea();
InsertarPalabraEnLinea (2, 1, "Palabra1");
InsertarPalabraEnLinea (2, 2, "Palabra2");
ImprimirTexto();
```

Salida:

```
1:
2: Palabra1 Palabra2

InsertarPalabraEnLinea (2, 1, "Palabra3");
ImprimirTexto();
```

Salida:

```
1:
2: Palabra3 Palabra1 Palabra2

InsertarPalabraYDesplazar(2, 2, "Palabra4");
ImprimirTexto();
```

Salida:

```
1:
2: Palabra3 Palabra4 Palabra 1
3: Palabra2

InsertarPalabraYDesplazar (2, 3, "Palabra5");
ImprimirTexto();
```

Salida:

1:
2: Palabra3 Palabra4 Palabra5
3: Palabra1 Palabra2

3.9. Borra la palabra en la posición indicada.

Firma: `Retorno borrarPalabra(int numContactoOrigen, int numMensaje, int posicionLinea, int posicionPalabra);`

Descripción: 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.

Retornos posibles	
OK	Se pudo insertar palabra con éxito.
ERROR	En caso de que no exista el número de contacto o que no exista el número de mensaje para el contacto indicado. 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.

Ejemplo:

```
InsertarLinea();
InsertarLinea();
InsertarPalabraEnLinea (2, 1, "Palabra1");
InsertarPalabraEnLinea (2, 2, "Palabra2");
InsertarPalabraEnLinea (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

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

Firma: `Retorno borrarOcurrenciasPalabraEnLinea(int numContactoOrigen, int numMensaje, int posicionLinea, String palabraABorrar);`

Descripción: 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.

Retornos posibles	
OK	Si se pudieron borrar las ocurrencias de la palabra con éxito.
ERROR	En caso de que no exista el número de contacto o que no exista el número de mensaje para el contacto indicado. Si se pudieron borrar las ocurrencias de la palabra con éxito.
NO_IMPLEMENTADA	Cuando aún no se implementó. Es el tipo de retorno por defecto.

es válida solamente si posicionLinea existe en el texto.

```
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
```

3.11. Imprime la línea por pantalla.

Firma: `Retorno imprimirLinea(int numContactoOrigen, int numMensaje, int posicionLinea);`

Descripción: 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.

Retornos posibles	
OK	S Si se mostró la línea.
ERROR	En caso de que no exista el número de contacto o que no exista el número de mensaje para el contacto indicado. Si la posición de la línea no es válida
NO_IMPLEMENTADA	Cuando aún no se implementó. Es el tipo de retorno por defecto.

Ejemplo:

```
InsertarLinea();
InsertarLinea();
ImprimirLinea(2);
```

Salida:

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

Salida:

2: Palabra1 Palabra2

OPERACIONES RELATIVAS AL DICCIONARIO:

3.12. Agrega una palabra al diccionario.

Firma: `Retorno ingresarPalabraDiccionario(String palabraAIngresar);;`

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

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.

Ejemplo:

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

Salida:

```
Bosque
Hoja
Hojalata
```

Nota: ImprimirDiccionario se decribe en el punto 3.14.

3.13. Borra una palabra del diccionario.

Firma: `Retorno borrarPalabraDiccionario(Cadena palabraABorrar);`

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

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.

Ejemplo:

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

Salida:

Bosque
Hojalata

3.14. Muestra las palabras del diccionario alfabéticamente.

Firma: `Retorno imprimirDiccionario();`

Descripción: 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".

Retornos posibles	
OK	Se mostro diccionario correctamente
ERROR	No hay errores
NO_IMPLEMENTADA	Cuando aún no se implementó. Es el tipo de retorno por defecto.

Ejemplo:

ImprimirDiccionario();

Salida:

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

Salida:

Bosque
Hoja
Hojalata

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

Firma: `Retorno ImprimirTextoIncorrecto();`

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

Retornos posibles	
OK	Se muestra el texto
ERROR	No hay errores
NO_IMPLEMENTADA	Cuando aún no se implementó. Es el tipo de retorno por defecto.

```
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
```

Salida:

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

Salida:

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

4. Reportes

4.1. Muestra la cantidad de mensajes enviados por un contacto

Firma: `Retorno cantidadDeMensajes(int numContactoOrigen);`
Descripción: Muestra en una matriz la cantidad de mensajes que fueron enviados por día desde un contacto indicado.

Retornos posibles	
OK	Se muestra el texto con los mensajes envidados por día.
ERROR	En caso de que el contacto no exista.
NO_IMPLEMENTADA	Cuando aún no se implementó. Es el tipo de retorno por defecto.

Ejemplo:

	25-Ene	28-Feb	3-Mar	10-Jul	14-Jul
Micaela	23	4	1	34	22
Felipe	34	4	33	1	92
Romina	4	3	3	2	4
Camila	22	0	2	1	3
Marco	1	0	34	5	11
Diego	3	0	2	2	1

5. Documentación

Se pide que la documentación incluya:

1. Las pre y post condiciones de los métodos solicitados
2. Diagrama de la estructura de datos que se implementó para representar el sistema de reservas junto con una breve explicación indicando por qué eligió dichas estructuras
3. El conjunto de pruebas diseñadas y ejecutadas y el resultado obtenido de esta ejecución con un screenshot de la pantalla mostrando el resultado de cada uno de los tests.
4. Los nombres de cada @Test (caso de prueba) debe ser descriptivos, dejando claro que es lo que se pretende testear. No se aceptará Tests con nombres Test1, Test2, etc.

Facultad de Ingeniería
Bernard Wand-Polak
Cuareim 1451
11.100 Montevideo, Uruguay
Tel. 2902 15 05 Fax 2908 13 70
www.ort.edu.uy

Información importante

- Puntaje mínimo/máximo: 0/40
- Los obligatorios se realizan por equipos de hasta 3 **estudiantes.**
- Plazo máximo de segunda entrega (con boleta):
- Se deberán respetar los formatos de impresión dados para las operaciones que imprimen en consola.
- El resto de las operaciones no deben imprimir nada en consola.
- El sistema no debe requerir ningún tipo de interacción con el usuario por consola.
- Es obligación del estudiante mantenerse al tanto de las aclaraciones que se realicen en clase o a través del foro de aulas.
- **Se la selección adecuada de las estructuras para modelar el problema y la eficiencia en cada una de las operaciones es muy importante.**
- Deberá aplicar la metodología vista en el curso.
- Para la presentación de la documentación se publicará en aulas.ort.edu.uy un template. (El uso de este template es obligatorio).
- El proyecto será implementado en lenguaje JAVA sobre una interfaz que se publicará en el sitio de la materia en aulas.ort.edu.uy (El uso de esta interfaz es obligatorio).

Facultad de Ingeniería
Bernard Wand-Polak
Cuareim 1451
11.100 Montevideo, Uruguay
Tel. 2902 15 05 Fax 2908 13 70
www.ort.edu.uy

RECORDATORIO: IMPORTANTE PARA LA ENTREGA

- **Obligatorios**

La entrega de los obligatorios será en formato digital online, a excepción de algunas materias que se entregarán en Bedelía y en ese caso recibirá información específica en el dictado de la misma.

Los principales aspectos a destacar sobre la **entrega online de obligatorios** son:

1. Ingresá al sistema de Gestión.
2. En el menú, seleccioná el ítem "Evaluaciones" y la instancia de evaluación correspondiente, que figura bajo el título "Inscripto".
3. Para iniciar la entrega hacé clic en el ícono:
4. Ingresá el número de estudiante de cada uno de los integrantes y hacé clic en "Agregar". El sistema confirmará que los integrantes estén inscriptos al obligatorio y, de ser así, mostrará el nombre y la fotografía de cada uno de ellos. Una vez agregados todos los integrantes, hacé clic en "Crear equipo".

   Cualquier integrante podrá:

   • Modificar la integración del equipo.

   • Subir el archivo de la entrega.

5. Seleccioná el archivo que deseás entregar. Verificá el nombre del archivo que aparecerá en la pantalla y hacé clic en "Subir" para iniciar la entrega. Cada equipo (hasta 2 estudiantes) debe entregar **un único archivo en formato zip o rar** (los documentos de texto deben ser pdf, y deben ir dentro del zip o rar). El archivo a subir debe tener **un tamaño máximo de 40mb**

   Cuando el archivo quede subido, se mostrará el nombre generado por el sistema (1), el tamaño y la fecha en que fue subido.
6. El sistema enviará un e-mail a todos los integrantes del equipo informando los detalles del archivo entregado y confirmando que la entrega fue realizada correctamente.
7. Podés cerrar la pestaña de entrega y continuar utilizando Gestión o salir del sistema.
8. La **hora tope para subir el archivo será las 21:00** del día fijado para la entrega.
9. La entrega se podrá realizar desde cualquier lugar (ej. hogar del estudiante, laboratorios de la Universidad, etc).
10. Aquellos de ustedes que presenten alguna dificultad con su inscripción o tengan inconvenientes técnicos, por favor contactarse con el Coordinador o Coordinación adjunta **antes de las 20:00hs.** del día de la entrega, o enviando mail a las tres siguientes direcciones: gervaz@ort.edu.uy ,alamon@ort.edu.uy y terra@ort.edu.uy, o telefónicamente al 29021505 - int 1156 (de 8:00 a 20:00 hs) .

Si tuvieras una situación particular de fuerza mayor, debes contactarte con suficiente antelación al plazo de entrega, al Coordinador de Cursos (gervaz@ort.edu.uy) o Secretario Docente (paulos@ort.edu.uy).