TRABAJO OBLIGATORIO PRIMERA PARTE
SIMULADOR DE UN MANEJADOR DE BASE DE DATOS

Se desea implementar un sistema manejador de base de datos que permita crear y mantener tablas, almacenar datos en ellas y realizar ciertas operaciones sobre los datos.

Una **base de datos** es un conjunto de tablas que poseen nombres únicos que permiten identificarlas. Una **tabla** es, desde el punto de vista lógico, un conjunto de **filas** y **columnas** (**campos**) donde se distribuye la información.

Ejemplo:

Tabla Personas

Nombre	**Apellido**	**CI**
María	Castaña	0.000.001
Juan	Pérez	2.256.698
Daniel	Rodríguez	3.333.444
Laura	Pérez	2.123.328

Cada columna tiene un nombre que debe ser único dentro de las columnas de la tabla y contiene datos de una misma naturaleza. Si tomamos como ejemplo la tabla de personas, cada columna representa un atributo de las personas. Luego las filas contienen los datos de cada persona. Una **tupla** es la colección de datos presentes en una fila de la tabla, considerando el orden en que se presentan los datos. Esto nos lleva a que dos tablas con todas sus columnas iguales pero dispuestas en distinto orden se consideren tablas distintas. En otras palabras, una tabla es un conjunto de tuplas y una tupla es un conjunto de datos, debiendo cumplirse que para una misma tabla todas las tuplas tienen la misma cantidad de datos de los mismos tipos y en el mismo orden. No se permite que en una tabla existan dos tuplas idénticas.

Existe, para este trabajo, una sola forma de calificar a una columna de una tabla: PRIMARY KEY. El calificador PRIMARY KEY se aplica a una y sólo una columna de una tabla. PRIMARY KEY indica que los valores de la columna no pueden ser vacíos y son únicos. Así, se puede identificar unívocamente a un tupla por el valor del campo correspondiente a la columna PRIMARY KEY. Notar que en el ejemplo previo de la tabla de personas, Apellido no podría ser un campo PRIMARY KEY, pero sí el campo CI (de acuerdo a las tuplas del ejemplo).

Por motivos de simplicidad consideramos,

1) sólo dos tipos de datos almacenables en una tabla: string e integer. Un string es una secuencia de caracteres donde se excluyen los siguientes caracteres: el mayor (>), el menor (<), el igual (=), el distinto (!), el dos puntos (:) y el asterisco (*).
2) que la primera columna, y sólo ésta, de todas las tablas tendrá el calificador PRIMARY KEY.

Los calificadores serán por tanto manejados implícitamente en este trabajo, teniendo en cuenta las posiciones de las columnas en las tablas.

En resumen, el sistema deberá poder almacenar y administrar tablas que cumplan el siguiente esquema:

- Un nombre que identifica a la tabla de manera única

- Las columnas, de las que se conoce su:
  - Nombre (que la identifica dentro de la tabla)
  - Tipo de datos (string, integer)
  - Calificador, implícito, que se aplica a esa columna: PRIMARY KEY o ninguno.

Consideraciones Generales

Tipo de datos a manejar:

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

Pueden definirse tipos de datos auxiliares.

Toda operación del sistema debe retornar un elemento de tipo **`TipoRet`**. Si la operación se realizó exitosamente, deberá retornar `OK`; si la operación no se pudo realizar de forma exitosa deberá retornar `ERROR` e imprimir ***un mensaje de error correspondiente al error producido***; y finalmente, si la operación no fue implementada, deberá retornar `NO_IMPLEMENTADA`. En cualquier caso que la ejecución de una operación no sea satisfactoria (retorne ERROR), el estado del sistema deberá permanecer inalterado.

Al comenzar la ejecución del sistema se tendrá **una** base de datos vacía, sin tablas.

El sistema debe permitir realizar las operaciones que detallamos a continuación directamente sobre la línea de comandos, no utilizando un menú.

Operaciones sobre la Base de Datos

OPERACIONES PARA MODIFICAR LA BASE DE DATOS:

CREAR TABLA – createTable(nombreTabla)

Descripción: Crea una nueva tabla vacía (sin columnas ni tuplas) en la base de datos con nombre *nombreTabla*, siempre que no exista ya una tabla con dicho nombre.

`Ejemplo. Si se desea crear 2 tablas llamadas Personas y Productos:`

```
createTable (Personas)
createTable (Productos)
```

```
TipoRet createTable (char * nombreTabla);
```

Retornos posibles:	
OK	• Si se pudo ejecutar exitosamente el comando.
ERROR	• Si *nombreTabla* existe.
NO_IMPLEMENTADA	• Cuando aún no se implementó. Es el tipo de retorno por defecto.

ELIMINAR TABLA – dropTable(nombreTabla)

Descripción: Elimina de la base de datos la tabla de nombre *nombreTabla*, y las tuplas que la misma posee, si *nombreTabla* existe.

```
Ejemplo. Eliminar la tabla Productos:

dropTable (Productos)
```

```
TipoRet dropTable (char * nombreTabla);
```

Retornos posibles:	
OK	• Si se pudo ejecutar exitosamente el comando.
ERROR	• Si *nombreTabla* no existe.
NO_IMPLEMENTADA	• Cuando aún no se implementó. Es el tipo de retorno por defecto.

OPERACIONES PARA MODIFICAR UNA TABLA:

– addCol (nombreTabla, nombreCol)

Descripción: Agrega a la tabla de nombre *nombreTabla*, si ésta existe, una nueva columna al final de nombre *nombreCol*, si ésta no existe. Si la tabla tiene tuplas la operación no resultará válida. Si esta es la primera columna que se agrega a la tabla su calificador será PRIMARY KEY. El tipo de datos de la nueva columna es string.

```
Ejemplo. Crear 3 columnas en la tabla Personas llamadas CI, Apellido
y Nombre:

addCol (Personas,CI)
addCol (Personas,Apellido)
addCol (Personas,Nombre)
```

Tabla Personas *(con CI PRIMARY KEY)*

CI	**Apellido**	**Nombre**

```
TipoRet addCol (char * nombreTabla, char * NombreCol)
```

Retornos posibles:

OK	• Si se pudo ejecutar exitosamente el comando.
ERROR	• Si *nombreTabla* no existe. • Si *nombreCol* existe. • Si la tabla *nombreTabla* tiene al menos una tupla.
NO_IMPLEMENTADA	• Cuando aún no se implementó. Es el tipo de retorno por defecto.

– dropCol (nombreTabla, nombreCol)

Descripción: Elimina de la tabla de nombre *nombreTabla*, si ésta existe, la columna de nombre *nombreCol*, si ésta existe. Si la tabla tiene tuplas, entonces se eliminará de éstas el campo correspondiente a la columna eliminada. Si la tabla posee una única columna de nombre *nombreCol* entonces quedará como tabla vacía. Si la columna a eliminar es la PRIMARY KEY, la operación resultará inválida, salvo que ésta sea la única columna de la tabla, en cuyo caso quedará como tabla vacía, como se describió previamente.

Ejemplo. Eliminar la columna Apellido de la tabla Personas:

```
dropCol (Personas,Apellido)
```

Tabla Personas

CI	Nombre

TipoRet dropCol (char * nombreTabla, char * NombreCol)

Retornos posibles:	
OK	• Si se pudo ejecutar exitosamente el comando.
ERROR	• Si *nombreTabla* no existe. • Si *nombreCol* no existe. • Si *nombreCol* es la primera columna y la tabla tiene más columnas.
NO_IMPLEMENTADA	• Cuando aún no se implementó. Es el tipo de retorno por defecto.

Operaciones para la Edición de Datos

INSERTAR TUPLA – insertInto(nombreTabla, valoresTupla)

Descripción: Inserta en la tabla de nombre *nombreTabla*, si ésta existe, una tupla con los valores dados en *valoresTupla* para las columnas. La correspondencia de los valores con las columnas definidas, en la habitualmente llamada *metadata*, es posicional. Los valores de *valoresTupla* se separan con el uso del carácter dos puntos (:).
Si no se indican todas las columnas la operación resultará inválida. También será inválida si no hay al menos una columna en la tabla *nombreTabla* o si la tupla a insertar pertenece a la tabla o si el valor de la columna PRIMARY KEY ya existe.

Ejemplo. Insertar tuplas en la tabla Personas:

```
insertInto (Personas,3333111:Telma)
insertInto (Personas,2566499:Jose)
insertInto (Personas,4232323:Juan)
insertInto (Personas,1555000:Pepe)
insertInto (Personas,2565000:Maria)
```

Tabla Personas

CI	**Nombre**
3333111	Telma
2566499	Jose
4232323	Juan
1555000	Pepe
2565000	Maria

`TipoRet insertInto (char * nombreTabla, char * valoresTupla)`

Retornos posibles:	
OK	• Si se pudo ejecutar exitosamente el comando.
ERROR	• Si *nombreTabla* no existe. • Si la cantidad de campos en valoresTupla no coincide con la cantidad de columnas de la tabla *nombreTabla*. • Si la tabla *nombreTabla* no tiene columnas. • Si la tupla ya existe o si el valor de la primera columna se repite en otra tupla.
NO_IMPLEMENTADA	• Cuando aún no se implementó. Es el tipo de retorno por defecto.

ELIMINAR TUPLA – deleteFrom(nombreTabla, condicionEliminar)

Descripción: Elimina de la tabla de nombre *nombreTabla*, si éste existe, todas las tuplas que cumplen la condición *condiciónEliminar*. En caso de que la condición sea "", se eliminan todas las tuplas de la tabla. Si ninguna tupla cumple la condición, la operación no tendrá efecto.
El formato de las condiciones es: *columna operador valor* (sin espacios en blanco intermedios). Los operadores a utilizar son: = "igual", ! "Distinto", > "Mayor" y < "Menor". Por ejemplo, Sexo=Masculino, Edad!20, Apellido>Perez. Para comparar strings con el operador > o < se utilizará el orden lexicográfico habitual.
El operador * "Igual Prefijo": una condición que contiene el operador * resulta verdadera para una tupla si, y sólo si, el valor de la columna en dicha tupla empieza con el prefijo especificado.
Por ejemplo, CI*2256 (Todas las cédulas que comienzan con este número), Apellido*Per (Todos los apellidos que comienzan con Per; Ej: Perez, Peralta).

```
Ejemplo. Borrar tuplas de la tabla Personas:

deleteFrom (Personas,Nombre=Jose)
```

```
deleteFrom (Personas,Nombre=Maria)
```

Tabla Personas

CI	**Nombre**
3333111	Telma
4232323	Juan
1555000	Pepe

`TipoRet deleteFrom (char * nombreTabla, char * condicionEliminar)`

Retornos posibles:	
OK	• Si se pudo ejecutar exitosamente el comando.
ERROR	• Si *nombreTabla* no existe. • Si la columna dentro de *condicionEliminar* no pertenece a la tabla *nombreTabla*. No se considera error si ninguna tupla cumple la condición. Además se asume que la condición *condicionEliminar* respeta el formato establecido.
NO_IMPLEMENTADA	• Cuando aún no se implementó. Es el tipo de retorno por defecto.

MODIFICAR TUPLA - UPDATE(NOMBRETABLA, CONDICIONMODIFICAR, COLUMNAMODIFICAR, VALORMODIFICAR)

Descripción: Modifica en la tabla de nombre nombreTabla, si éste existe, el valor de las tuplas en la columna de nombre columnaModificar, si éste existe, que cumplen la condición condiciónModificar. En la columna especificada de las tuplas que cumplen la condición se asigna el valor valorModificar, siempre que este valor sea del tipo adecuado y satisfaga el calificador de la columna especificada. La condición respeta el formato descrito para condiciones.

Ejemplo: Modificar la CI de Pepe.

```
update (Personas,Nombre=Pepe,CI,1555000);
```

Tabla Personas

Nombre	**CI**
Telma	3333111
Juan	4232323
Pepe	1555000

Operaciones de Impresión

LISTAR TABLA – printDataTable (nombreTabla)

Descripción: Imprime las tuplas de la tabla de nombre *nombreTabla*, si ésta existe. Los nombres y los valores de las columnas se expresan en el formato $columna_1$:$columna_2$: ... :$columna_n$. Las tuplas se muestran ordenadas ascendentemente por la PRIMARY KEY. Primero se imprime el nombre de la tabla, luego los nombres de las columnas, separados con (:), y por último las tuplas, cuyos campos se separan también con (:).

Ejemplo. Si queremos ver las tuplas de la tabla Personas:

```
printDataTable (Personas)
```

```
Personas
CI:Nombre
1555000:Pepe
3333111:Telma
4232323:Juan
```

TipoRet printdatatable (char * NombreTabla)

Retornos posibles:	
OK	• Si se pudo ejecutar exitosamente el comando.
ERROR	• Si *nombreTabla* no existe. No es error si no hay columnas o tuplas en *nombreTabla*. En este caso deberá imprimir "no hay tuplas en *nombreTabla*".
NO_IMPLEMENTADA	• Cuando aún no se implementó. Es el tipo de retorno por defecto.

COMANDOS EJECUTADOS ERRÓNEAMENTE

Aunque lo siguiente fue expresado al comienzo de este documento, decidimos sintetizar aquí la política de manejo de errores frente a la ejecución de las operaciones del sistema. Cada operación tiene determinadas precondiciones para que funcione correctamente. En caso de que alguna de estas precondiciones no se cumpla la operación debería retornar ERROR (tal cuál se señala para cada operación), imprimiendo además por pantalla un texto breve apropiado a cada caso para destacar el tipo de error ocurrido. En cualquier caso que la ejecución de una operación no sea satisfactoria (retorne ERROR), el estado del sistema deberá permanecer inalterado.

SOBRE LOS CHEQUEOS

Se permite considerar que la sintaxis de la entrada que recibirá el programa es válida. Esto quiere decir que no se requiere la realización de chequeos como los siguientes:

- cadenas de nombres de columnas y valores de tuplas que no respeten el formato establecido (usando el (:) como separador).

- nombres de tablas y columnas que tengan caracteres no permitidos.
- condición en la operación deleteFrom que no respete el formato establecido.

Esto no quiere decir que no se deban chequear las condiciones establecidas para las operaciones en la letra del obligatorio. Por ejemplo, creación de una tabla ya existente, supresión de una columna inexistente, entre otros.

Operaciones obligatorias	**Operaciones Opcionales**
Operaciones sobre la Base de Datos Crear Tabla Eliminar Tabla	Operaciones sobre la Base de Datos Modificar Tabla
Operaciones para la Edición de Datos Insertar Tupla Eliminar Tupla	Operaciones para la Edición de Datos Modificar Tupla
Operaciones para la Impresión de Información Listar Tabla	

.

ENTREGA Y FORMA DE TRABAJO

El obligatorio debe entregarse por correo electrónico al docente del curso **antes del 17/10 a las 23:59**.

Los trabajos pueden hacerse en grupos de hasta 2 estudiantes.