EJERCICIO

SQL

Se poseen las siguientes tablas:

USUARIOS	FAVORITOS	USUARIOSPAGOS	PAGOS
codigousuario usuario clave edad	codigousuario codigousuariofavorito	codigopago codigousuario	codigopago importe fecha

Aclaración: la tabla **"favoritos"** une usuarios con otros usuarios a través de los campos **codigousuario** y **codigousuariofavorito.** Un ejemplo seria que si el usuario con código 1 tiene de favoritos a los usuarios de código 5,6 y 8 en la tabla se debe mostrar:

CODIGOUSUARIO	CODIGOUSUARIOFAVORITO
1 1 1	5 6 8

a. Encuentre el error en la siguiente consulta, si la intención era devolver todos los nombres de usuario (campo "usuario") y clave de los usuarios que tengan 18 años, ordenados alfabéticamente por el nombre de usuario

SELECT USUARIO, CLAVE FROM USUARIOS ORDER BY CODIGOUSUARIO

b. Encuentre el error en la consulta, que debería devolver todos los usuarios con sus favoritos, únicamente mostrando nombres de usuario: en la primera columna que se muestre el nombre de usuario del dueño del favorito, y en la segunda que aparezca el nombre de usuario del favorito. Si le parece necesario, reescribir la consulta

```
SELECT USUARIOS.USUARIO, FAVORITOS.CODIGOUSUARIO
FROM USUARIOS
INNER JOIN FAVORITOS ON FAVORITOS.CODIGOUSUARIO = USUARIOS.CODIGOUSUARIO
```

c. Indique qué índices (sean clave primaria o no) debería tener cada tabla, y por qué.

d. ¿Existe algo de la estructura de la base de datos que cree se debería modificar? ¿Por qué?

EJERCICIO ALGORÍTMICO:

ORDENAR

Encuentre el o los errores en la función, si la idea es ordenar un vector de N dimensiones, por cualquiera de ellas, en forma ascendente. Los parámetros deben ser primero el vector, y luego la dimensión ordinal por el cual ordenar. La función debe devolver un nuevo vector ya ordenado.

Ejemplo:

```
v[0] = array("nombre"  => "Juan", "apellido"  => "Perez", "edad"  => 26);
v[1] = array("nombre"  => "Carlos", "apellido"  => "Gomez", "edad"  => 30);
v[2] = array("nombre"  => "Ernesto", "apellido"  => "Ramirez", "edad"  => 22);
```

Llamando a la función con los parámetros $v y "edad", la misma debería devolver el listado ordenado por la edad (en forma ascendente), es decir:

```
w[0] = array("nombre"  => "Ernesto", "apellido"  => "Ramirez", "edad"  => 22);
w[1] = array("nombre"  => "Juan", "apellido"  => "Perez", "edad"  => 26);
w[2] = array("nombre"  => "Carlos", "apellido"  => "Gomez", "edad"  => 30);
```

FUNCION:

```
function orderArray($array, $dimension){
        $length = count($array);
        for ($outer = 0; $outer < $length; $outer++) {
                for ($inner = 0; $inner < $length; $inner++) {
                        if ($array[$outer] < $array[$inner]) {
                                $tmp = $array[$outer];
                                $array[$outer] = $array[$inner];
                                $array[$inner] = $tmp;
                        }
                }
        }
}
```

EJERCICIO ALGORÍTMICO:

RECURSIVIDAD

Encuentre el o los errores en la función, que debería recorrer todos los archivos existentes en el directorio "/files/" recursivamente, incluyendo los archivos de todos los subdirectorios. Una vez concluido el recorrido, la función debería devolver un vector que contenga todos los archivos (indicando la ruta completa).

Ejemplo:

Vec[0] = “/files/config.inc”
Vec[1] = “/files/connection.php”
Vec[2] = “/files/images/logo.gif”
Vec[3] = “/files/images/ads/160x600.gif”

FUNCION:

```
$main = "/files/";

function readDirs($main){
	$dirHandle = opendir($main);
	while($file = readdir($dirHandle) {
		if(is_dir($main . $file) && $file != '.' && $file != '..') {
			$vec = readDirs($file);
		} else {
			$vec[] = $main.$file;
		}
	}
}
```

Ejercicio Framework

LARAVEL

1. Configuración de un proyecto en el Framework Laravel el cual deberá publicarse en un gitlab o github publico que contenga las siguientes características:
    a. Registro, autenticación y recuperación de contraseña para los datos de pacientes en un sistema de clínicas.
    b. Una vez autenticado puede visualizar el directorio médico.
    c. En el administrador, generar un módulo CRUD para la gestión de los datos del paciente tomar en cuenta: datos mínimos requeridos para la atención de pacientes, manejo de tipo de documentos, eliminación lógica.
    d. En el administrador, generar un módulo que permita gestionar Especialidad y Médicos, el módulo debe permitir: gestionar el CRUD, eliminación lógica, que lo médicos pueden tener más de una especialidad asociada y demás consideraciones básicas para el caso.
    e. Se debe gestionar en Gitlab o GitHub un repositorio público con el modelo de base de datos y aplicación con el Framework Laravel, está permitido utilizar paquetes y plugins de apoyo en el proyecto.