***Lea los problemas a continuación. Resuelva lo que se solicita usando las tecnologías indicadas. Puede resolver este documento en el orden que mejor le parezca.***

**Sección práctica**

**Problema 1 (Extra)**

Su empresa necesita una demo en HTML y JavaScript que realice las siguientes tareas:

1. Muestre un campo de texto al usuario. El campo puede recibir hasta 15 caracteres de cualquier tipo (alfanuméricos y signos especiales).
2. El campo de texto debe llevar el identificador “MENSAJE” (puede ser minúsculas o mayúsculas).
3. Muestre el contenido del campo de texto en una ventana emergente del navegador cuando se cumpla la siguiente condición:
   1. La longitud del texto ingresado es igual a 15 caracteres.

NOTAS:

1. Separe el código JavaScript en un archivo llamado “problema1.js”
2. El archivo HTML debe llamarse “index.html”. Asegúrese de usar la estructura base de HTML5.
3. Evite dejar mensajes propios en la consola de JavaScript; no se preocupe de mensajes que el navegador coloque.

**Problema 2**

Se le ha solicitado desarrollar un programa de consola en C# o Java. Utilizando el IDE de su preferencia, diseñe el programa para que realice las siguientes tareas:

1. Lea un valor N del usuario a través de la consola. Verifique que el valor N ingresado corresponde a un número mayor o igual a cero y menor a 15.
2. Determine de forma recursiva los primeros N números de la secuencia de Fibonacci e imprima en pantalla.

**EJEMPLO 1**

***INPUT*:** 10

***OUTPUT*:** 0, 1, 1, 2, 3, 5, 8, 13, 21, 34,

**EJEMPLO 2**

***INPUT*:** 4

***OUTPUT*:** 0, 1, 1, 2,

**Problema 3**

Un cambio en el requerimiento del cliente le solicita clonar el proyecto anterior y modificarlo para que realice la siguiente tarea:

1. Lea un valor N del usuario a través de la consola. Verifique que el valor N ingresado corresponde a un número mayor o igual a cero y menor o igual a quince.
2. Determine de forma recursiva la sumatoria de los primeros N números de la secuencia de Fibonacci e imprima el resultado en pantalla.

**EJEMPLO 1**

***INPUT*:** 10

***OUTPUT*:** 88

**EJEMPLO 2**

***INPUT*:** 4

***OUTPUT*:** 4

**Problema 4**

Cree un nuevo proyecto en su IDE preferido, para diseñar un programa de consola en C# o Java. El programa debe realizar las siguientes tareas:

1. Lea un valor N del usuario a través de la consola. Verifique que el valor N ingresado corresponde a un número mayor o igual a cero y menor o igual a quince.
2. Construya un matriz de N\*N.
3. Rellene cada celda de la matriz de forma que obtenga los resultados del ejemplo.
4. No es necesario que imprima la matriz.

**EJEMPLO 1**

***INPUT*:** 3

|  |  |  |
| --- | --- | --- |
| 0 | 1 | 2 |
| 1 | 2 | 3 |
| 2 | 3 | 4 |

***OUTPUT*:**

***INPUT*:** 4

|  |  |  |  |
| --- | --- | --- | --- |
| 0 | 1 | 2 | 3 |
| 1 | 2 | 3 | 4 |
| 2 | 3 | 4 | 5 |
| 3 | 4 | 5 | 6 |

***OUTPUT*:**

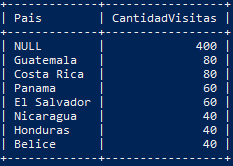
**Problema 5**

*Nota: el siguiente problema requiere contar con el Shell de MySQL instalado en su computador. Sino cuenta con este programa, puede ingresar usando un navegador web u otro cliente con soporte para MySQL, pero no se recomienda este método ya que es mucho más lento.*

Inicie sesión en MySQL. Se le ha dado acceso a una base de datos llamada ***“pabloalvarado”***. La misma cuenta con una tabla llamada ***“registoPrueba”***. Escriba una consulta SQL que realice lo siguiente:

1. Muestre 2 columnas “Pais” y “CantidadVisitas”:
   1. La columna Pais mostrará el nombre de cada región.
   2. La columna “CantidadVisitas” es la sumatoria total de registros de cada región en la tabla “registoRegiones”.
2. Calcule la sumatoria total de registros de la consulta.
3. Ordene primero la columna “CantidadVisitas” y por último ordene la columna “Pais”, ambas de forma descendente.

Debería de ver el siguiente resultado:



La estructura de la tabla “registroRegiones” es:

CREATE TABLE registoRegiones (

id int AUTO\_INCREMENT PRIMARY KEY,

nombre VARCHAR(30) NOT NULL

);

**Copie y pegue la consulta SQL aquí:**

**Sección teórica**

1. Un árbol binario es una estructura de datos que facilita ordenar valores. Insertar un nuevo valor en el árbol binario conlleva complejidad mientras que buscar un valor representa una complejidad . También existe una estructura de datos denomina lista enlazada. Insertar un valor en esta estructura tiene una complejidad *O(1)* pero buscar un nuevo valor implica en el peor escenario. Sabiendo esto responda:

¿Qué estructura usaría para implementar el registro de transacciones de un usuario en un sistema? Asuma, por el momento, que el sistema no es de consulta, sino solamente para guardar transacciones.

Implementaría una lista enlazada, ya que esto permitiría que el sistema se ejecute de una mejor manera al tomar un menor tiempo las incersiones.

Suponga que el sistema requiere guardar en un disco duro mecánico todos los registros. Además, ahora se realizan regularmente ciertas consultas aleatorias para verificar el estado del sistema ¿Consideraría implementar alguna de estas estructuras de nuevo? ¿Por qué?

Consideraría utliziar un árbol binario, ya que conforme aumentan la cantidad de datos, las consultas aleatorias realizadas pueden ocasionar tiempos de espera excesivos. Asimismo, para un posterior análisis convendría tener los datos ordenados en un árbol.

Nota: el árbol binario facilita ordenar valores, más no siempre significa que el árbol en sí mismo este balanceado. Muchos arboles pueden resultar siendo listas si el orden de los elementos ingresados tiene un comportamiento “ascendente” o “descendente” para la mayoría de sus valores, en lugar de que estos sean aleatorios. En este caso, el balanceo del árbol puede requerir múltiples reescrituras de los registros (que en este ejemplo se realizan sobre un disco duro mecánico) lo que podría causar lentitud del sistema. Sin embargo, emplear una lista tampoco sería una opción más viable, ya que las consultas aleatorias podrían ser demasiado lentas (similar al caso de un árbol que termina siendo una lista). Lo más adecuado sería analizar otras estructuras de datos que satisfagan estas necesidades de mejor manera, como un árbol B.

1. Las expresiones regulares son una forma muy eficiente de validar cadenas de texto. Construya una expresión regular que valide todos los aspectos siguientes:
   1. La cadena inicia con un carácter alfabético en minúscula seguido de N caracteres alfanuméricos para 0 ≤ N ≤ 15) y que termina siempre con la expresion “@bluedevs.com” (sin comillas) Cero o mas números

C → a – z

N → 0 – 15

T → @bluedevs.com

ER: CN\*T

Comentario: Tanto las Expresiones Regulares como las Gramáticas Regulares permiten definir Lenguajes Regulares. En este caso, si se utilizaba una Gramática Regular, una posible solución sería la siguiente (aunque extremadamente larga)

P = {

S → A

A → BCD

A → BD

B → (asuma todas las producciones de los símbolos abcdefghijklmnopqrstuvwxyz)

C → (asuma todas las producciones de N símbolos en Σ excepto “@” y “.”)

D → @bluedevs.com

}

La misma gramática puede representarse por una expresión regular como la siguiente:

ER = [a-z]+[A-z0-9]\*@bluedevs.com

b. La cadena solo contiene dígitos del 1 al 8. Al menos un numero

N → 1, 2, 3, 4, 5, 6, 7, 8

ER: N+

Comentario: Al igual que el caso anterior, existe una solución usando Gramticas Regulares y otra usando Expresiones Regulares. En caso de la expresión regular, la solución sería

ER = [1-8]+