Prueba Técnica de Inmersión: Desarrollo de Software - EmasT

Duración total: 2 horas

Instrucciones Generales

- Lea cuidadosamente todas las instrucciones antes de comenzar.
- Gestione su tiempo adecuadamente. Se proporcionan tiempos sugeridos para cada sección.
- Puede usar el lenguaje de programación de su preferencia para las secciones de codificación.
- Si tiene alguna pregunta, levante la mano y un supervisor se acercará a usted.

Parte 1: Ejercicios de Lógica de Datos (40 minutos)

Propósito: Evaluar la capacidad de manipulación de estructuras y pensamiento algorítmico.

1.1 El Elemento Solitario (10 minutos):

Dado un arreglo de números donde cada elemento se repite exactamente dos veces, excepto uno que aparece solo una vez, implemente una función que encuentre y retorne ese número único.

- *Ejemplo:* Entrada: [4, 1, 2, 1, 2] -> Salida: 4.

1.2 Limpieza de Ceros (10 minutos):

Dada una lista de números, mueva todos los ceros al final de la misma sin cambiar el orden relativo de los demás elementos.

- *Ejemplo:* Entrada: [0, 1, 0, 3, 12] -> Salida: [1, 3, 12, 0, 0].

1.3 Compresor de Texto (10 minutos):

Implemente un método para realizar una compresión básica de cadenas utilizando el recuento de caracteres repetidos. Si la cadena "comprimida" no es más pequeña que la original, debe devolver la original.

- *Ejemplo:* Entrada: aabcccccaaa -> Salida: a2b1c5a3.

1.4 El Número Faltante (10 minutos):

Se le entrega una lista que contiene $n-1$ números en el rango de 1 a n. No hay duplicados. Encuentre el número que falta en la secuencia.

- *Ejemplo:* Rango de 1 a 5, lista [1, 2, 4, 5] -> Salida: 3.

Parte 2: Ejercicio de Codificación Principal (65 minutos)

El Reto: Motor de Inventario y Reglas de EmasT

Desarrolle una aplicación simple de gestión de inventario utilizando principios de **Programación Orientada a Objetos (POO)** y diseño de **Base de Datos Relacional**.

Requisitos de Desarrollo:

1. **Jerarquía de Clases:** Cree una clase base Producto y dos clases derivadas: ProductoFisico y ProductoDigital.
2. **Encapsulamiento:** Proteja los datos de los ítems mediante modificadores de acceso adecuados.
3. **Lógica y Polimorfismo:** Implemente un método CalcularPrecioFinal() que aplique:
    - Si el producto es Digital, aplicar un **15% de descuento**.
    - Si el Stock actual es mayor a 50 unidades, aplicar un **5% de descuento adicional** (acumulable).
    - **Alerta:** Si el precio final es menor a 5.000, el sistema debe imprimir una advertencia de "Revisión de Margen Necesaria".

Requisitos de Base de Datos:

1. Diseñe un esquema simple para almacenar los productos y sus categorías.
2. Escriba consultas SQL para:
    - Listar todos los productos disponibles con stock menor a 10 unidades.
    - Calcular el promedio de precios de productos por cada categoría existente.

Parte 3: Diseño de Sistema (15 minutos)

Caso: Notificaciones EmasT

Diseñe un sistema simple para notificar a los usuarios cuando un producto físico vuelve a estar en stock.

1. Realice un **Diagrama de Clases UML** mostrando al menos 4 clases relevantes (ej. Producto, Usuario, Notificador, Suscripción).
2. Describa brevemente qué estrategia usaría para enviar 10.000 correos simultáneos sin bloquear la aplicación.

Criterios de Evaluación

- **Corrección y eficiencia:** Los algoritmos de lógica funcionan correctamente.
- **Principios POO:** Aplicación adecuada de herencia, encapsulamiento y polimorfismo.
- **Diseño de Datos:** Esquema de base de datos coherente y consultas SQL precisas.
- **Razonamiento Lógica:** Capacidad para estructurar soluciones a problemas nuevos.

¡Mucho éxito!