Apellidos Nombres	E01	E02	E03	E04	E05	E06	Nota

Paradigmas de Programación - UNLaM

Paradigma Imperativo - Orientado a Objetos - Java

1. Teórico
   a. Compare y contraste los conceptos de polimorfismo y sobrecarga de métodos en la programación orientada a objetos.
   b. Explique en qué situaciones es apropiado utilizar uno u otro en el diseño de un sistema.

2. Práctico
   a. Modelar con UML un programa que modele una red social básica. Cree clases para representar usuarios, publicaciones y comentarios. Asegúrese de utilizar conceptos como herencia, polimorfismo y encapsulamiento en su diseño.
   b. ¿Qué parte del sistema utiliza el polimorfismo? ¿Cómo?
   c. Proporcione el código fuente de al menos dos métodos relevantes para el funcionamiento de la red social, como por ejemplo `publicarMensaje` y `agregarComentario`.

Paradigma Lógico - Prolog

3. Teórico

Explicar los distintos tipos de asignaciones y comparaciones existentes en Prolog. Proporcionar ejemplos concretos para ilustrar su funcionamiento.

4. Práctico
   a. Diseñar una base de conocimientos en Prolog con información sobre diferentes libros, incluyendo su título, autor, género y cantidad de páginas.
   b. Luego, implementar reglas en Prolog que permitan al usuario realizar consultas y obtener información sobre los libros, tales como:
      i. `obtener_libros_autor(Autor, Libros)`: Obtiene libros escritos por el autor especificado.
      ii. `obtener_libros_con_casi_mas_largos(Libros)`: Obtiene libros que tienen tantas páginas como el segundo más largo.

Paradigma Funcional - Haskell

5. Teórico

Explique el concepto de listas por comprensión en Haskell y cómo se utilizan para construir listas de manera concisa y elegante. Describa cómo se definen las listas por comprensión y cómo se pueden aplicar filtros y transformaciones a través de ellas. Proporcione ejemplos concretos para ilustrar el uso de listas por comprensión.

6. Práctico

Implementar una función en Haskell que tome un número y devuelva una función que sume ese número a cualquier otro número. Utilice el currying para lograr esta funcionalidad.
Ejemplo:

```
sumar5 = sumarN 5
sumar5 10 -- retorna 15
```