CLASE 4

1. Tipos de Datos Algebraicos (TDA)

Son tipos de datos nuevos que tú defines a través de "constructores". La clave es que la forma de procesar estos datos se basa en identificar y manejar cada uno de sus constructores.

Se clasifican en:

- **Enumerativos**: Varios constructores, pero **sin argumentos**. Sirven para representar un conjunto de valores fijos (ej. data Direccion = Norte | Sur | Este | Oeste).
- **Registros (o Productos)**: Un **único constructor** con varios argumentos. Sirve para agrupar diferentes datos en una sola entidad (ej. data Persona = P String Int String).
- **Sumas (o Variantes)**: Varios constructores, y **algunos o todos tienen argumentos**. Permiten modelar datos que pueden tener distintas formas (ej. data Helado = Palito | Cucurucho Gusto).
- **Recursivos**: Son el foco principal. Son tipos que **se usan a sí mismos** en la definición de alguno de sus constructores. Esto permite crear estructuras de datos de tamaño indefinido.
    - **Ejemplos Lineales**: Listas , Pizza (data Pizza = Prepizza | Capa Ingrediente Pizza).
    - **Ejemplos Ramificados**: Árboles (data Tree a = EmptyT | NodeT a (Tree a) (Tree a)).

2. El Patrón Fundamental: Recursión Estructural

Es la técnica principal para trabajar con tipos de datos recursivos. La idea es que **la estructura de tu función debe reflejar la estructura del tipo de dato**.

Las reglas son simples y potentes:

1. **Un caso por cada constructor**: Tu función debe tener una ecuación o caso para cada constructor del tipo de dato.
2. **Llamada recursiva en partes recursivas**: En el caso del constructor recursivo, la función se llama a sí misma sobre los argumentos que son del mismo tipo.
3. **Combinar los resultados**: Tu única tarea es pensar cómo combinar los datos "nuevos" del constructor con el resultado de la llamada recursiva para obtener la solución final.

3. Aplicación de la Recursión Estructural

- **Sobre Listas**:
    - **Constructores**: [] (caso base) y (x:xs) (caso recursivo).
    - **Patrón de Función**:
        - f [] = (resultado para lista vacía).
        - f (x:xs) = (combina 'x' con el resultado de 'f xs').
- **Sobre Números Naturales**:
    - **Casos**: 0 (caso base) y n (caso recursivo, para n > 0).
    - **Patrón de Función**:

        - f 0 = (resultado para cero).
        - f n = (combina 'n' con el resultado de 'f (n-1)').
    - **¡Importante!**: Este patrón no funciona para números negativos.
- **Sobre Árboles Binarios (Tree a)**:
    - **Constructores**: EmptyT (caso base) y NodeT x t1 t2 (caso recursivo).
    - **Patrón de Función**:
        - f EmptyT = (resultado para árbol vacío).
        - f (NodeT x t1 t2) = (combina 'x' con los resultados de 'f t1' y 'f t2').

4. Puntos Clave para el Examen (Estrategia)

Cuando te enfrentes a un problema, sigue estos pasos:

1. **Identifica el Tipo de Dato**: ¿Es una lista, un árbol, un número o un TDA personalizado? Su definición te dará la estructura de la solución.
2. **Identifica los Constructores**: Esto te dirá cuántos casos necesitas en tu función.
3. **Define el Caso Base**: Resuelve el problema para los constructores no recursivos ([], EmptyT, Prepizza, etc.). Suele ser un valor trivial (ej. 0, [], True/False).
4. **Define el Caso Recursivo**:
    - Asume que la llamada recursiva (f xs, f (n-1), f t1, etc.) ya funciona y te da el resultado correcto para la parte más pequeña.
    - Pregúntate: ¿Qué tengo que hacer con el resto de los datos del constructor (x, n, ing, etc.) y el resultado de la llamada recursiva para construir la solución final?
5. **Presta atención a las Precondiciones**: Si un problema especifica PRECOND: i >= 0 o j >= i, puedes asumir que esos casos se cumplen y no necesitas manejarlos explícitamente.