Problema 1

Problema 1: 25%

- Escriba la reducción-β de la operación lógica NOT
- Escriba y explique como se vería la recursión y los ciclos.
- Explique cuando es prudente usar este tipo de programación y cuando no. De un ejemplo para cada caso.

reducción-β de operación lógica NOT

valor lógico	Definición Lambda
TRUE	$\lambda a . \lambda b . a$
FALSE	$\lambda a . \lambda b . b$
NOT	$\lambda p . p$ FALSE TRUE

reducción-β NOT TRUE

sustituir con definición:

(λ p. p FALSE TRUE) TRUE

TRUE FALSE TRUE

(λa. λb. a) FALSE TRUE

(λb. FALSE)

FALSE

NOT TRUE → FALSE

reducción-β NOT FALSE

(λ p. p FALSE TRUE) FALSE

FALSE FALSE TRUE

(λa. λb. b) FALSE TRUE

(λb. b)

TRUE

NOT FALSE → TRUE

Aplicar funciones a sus algoritmos sustituyendo variables.

recursión y ciclos en programación funcional

una función se llama así misma.

factorial(0) = 1
factorial(n) = n * factorial(n-1)

se ejecuta hasta conseguir un caso base, y luego las llamadas se van resolviendo en orden inverso

"ciclos"
↳ en lugar de bucles, utilizar funciones de orden superior

js { map, filter, reduce

sí usar programación funcional

cuando se utilizan colecciones de datos: listas, matrices, arreglos

programas fáciles de mantener

uso de paralelismo o concurrencia

evitar efectos secundarios o estados mutables

ejemplo: procesar una lista de compra
- acumular
- sumar total
- stock

cuando no

cuando modifican estructuras grandes por eficiencia

se programa dispositivos o sistemas cercanos al hardware

hay un control detallado del estado del programa o memoria

ejemplo: - Manejo de hardware bajo nivel
- Motores de juego optimizados
- programa memoria Muy limitada