

# UNIVERSIDAD DEL VALLE DE GUATEMALA

Teoría de Computación

Sección 40



## Laboratorio 12

Humberto Alexander de la Cruz Chanchavac - 23735

**GUATEMALA, 5 de noviembre de 2025**

## Laboratorio 12

### Problema 1: 25%

- Escriba la reducción- $\beta$  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.

### Problema 1

Reducción- $\beta$  para NOT

$$\text{not} = \lambda b. (\lambda b. \lambda x. \lambda y. b \ x \ y) \ b (\lambda x. \lambda y. y) (\lambda x. \lambda y. x)$$

$$= \lambda b. (\lambda b. \lambda x. \lambda y. b \ x \ y) \ b (\lambda x. \lambda y. y) (\lambda x. \lambda y. x)$$

$$= \lambda b (\lambda b \ \lambda x. \lambda y. b \ x \ y) \ b (\lambda x. \lambda y. y) (\lambda x. \lambda y. x)$$

$$= \lambda b. (\lambda x. \lambda y. b \ x \ y) (\lambda x. \lambda y. y) (\lambda x. \lambda y. x)$$

$$= \lambda b. (\lambda y. b (\lambda x. \lambda y. y) y) (\lambda x. \lambda y. x)$$

$$= \lambda b. b (\lambda x. \lambda y. y) (\lambda x. \lambda y. x)$$

$$= \text{not} = \lambda b. b \ \text{False} \ \text{True} //$$

**b)** En programación funcional no se usan bucles for o while, sino recursión

Ejemplo: función que cuenta hasta 3:

```
contar n =  
  if n == 3  
  then []  
  else n : contar (n + 1)
```

Si “n” es igual a 3, devuelve una lista vacía ([]), lo que actúa como condición base. En caso contrario, construye una lista agregando n al resultado de llamar nuevamente a la función con  $n + 1$ .

**c)** Cuando usarlo:

- Cuando trabajas con procesamiento de listas, árboles o transformaciones de datos puros, ejemplo:
  - Haskell
  - Lisp.
- No es ideal cuando se necesita mucha mutación de estado, ejemplo:
  - Juegos
  - Sistemas interactivos en tiempo real.