

Lenguajes de Programación

Examen Parcial 04 - 2025-1

Manuel Soto Romero Demian Alejandro Monterrubio Acosta Erik Rangel Limón
José Alejandro Pérez Márquez Dieter Tadeo García Rosas

19 al 22 de noviembre de 2024
Facultad de Ciencias UNAM

Ejercicio 1. (25 pts.) Evalúa las siguientes expresiones usando las técnicas de paso de parámetros que se indican. En cada caso debes mostrar cómo queda el ambiente y memoria final, tal y como se vio en clase.

(a) Evalúa el siguiente código bajo paso por valor y por referencia.

```
(let [(list1 (box '(1 2 3)))  
      (list2 (box '(4 5 6)))  
      (concat-lists (lambda (x y)  
                     (begin  
                       (set! x (append (unbox x) (unbox y)))  
                       (set! y '(0)))))]  
  
  (begin  
    (concat-lists list1 list2)  
    (list (unbox list1) (unbox list2))))
```

(b) Evalúa el siguiente código bajo paso por nombre y por necesidad.

```
(let [(acc 0)  
      (conditional (lambda (x)  
                     (if (> x 0)  
                         (begin (set! acc (+ acc 1)) acc)  
                         (begin (set! acc (- acc 1)) acc)))]  
      (compute (lambda (y) (* y y))))  
  
(compute (conditional acc)))
```

Ejercicio 2. (25 pts.) Modifica la siguiente función escrita en HASKELL para que aplique correctamente la técnica de memoización como vimos en clase. No puedes usar instrucciones que tengan efectos secundarios. ¿Qué hace la función? Justifica además como cambia la eficiencia en tiempo y en espacio con respecto a la definición original.

```
changeWays :: Int -> [Int] -> Int  
changeWays 0 _ = 1  
changeWays _ [] = 0  
changeWays amount (coin:coins)  
  | amount < 0 = 0  
  | otherwise = changeWays amount coins +  
                 changeWays (amount - coin) (coin:coins)
```

Ejercicio 3. (25 pts.) Realiza el juicio de tipo para cada una de las siguientes expresiones, usa las reglas vistas en clase o define una nueva regla en caso de ser necesario. Observa que la primera expresión no tiene anotaciones de tipo, por lo que tendrás que definir las reglas para verificar los tipos.

(a) `(let (g (lambda (x) (x 4)))
 (g (lambda (y) (- y 2))))`

(b) `(letrec (f : number -> number
 (fun (x : number) : number
 (if0 x 1 (* n (f (- n 1))))))
 (f 5))`

Ejercicio 4. (25 pts.) Realiza la inferencia de tipos vista en clase sobre la siguiente expresión, recuerda obtener las restricciones y usar el algoritmo de unificación para resolverlas.

`((lambda (x) (* x 2)) (+ 2 3))`