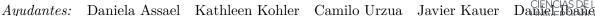
Lenguajes de Programación (2024/2)

Profesores: Feredico Olmedo Ismael Figueroa Auxiliares: Bastián Corrales Cristián Carrión





Auxiliar 6 Haskell y Scope

P1. Numeros naturales en Haskell

Considere la siguiente definicion inductiva de numeros naturales en Haskell:

```
data Natural = Zero | Succ Natural
```

Esto es equivalente en Racket a:

```
(deftype Natural (Zero) (Succ n))
```

- a) Implemente las funciones int2Nat :: Int -> Natural y nat2Int :: Natural -> Int que permiten transformar un entero en un Natural y viceversa.
- b) Implemente la función sumNatural :: Natural -> Natural -> Natural que dado dos Natural retorna el Natural correspondiente a la suma de ambos.
- c) Implemente la función fact Natural :: Natural -> Natural que dado un Natural entrega su factorial.

P2. Listas infinitas en Haskell Defina en Haskell lo siguiente:

- a) La función myRepeat :: a -> [a] que toma un elemento y crea una lista infinita con ese elemento. Ej: myRepeat 1 -> [1,1,1,1,1,1,...]
- b) La función myCicle :: [a] -> [a] que toma una lista y crea una lista infinita de repeticiones de dicha lista. Ej: myCycle [1,2,3] -> [1,2,3,1,2,3,1,2,3,...]

P3. Scope Estático vs Dinámico:

a) Para los siguientes programas (pertenecientes al lenguaje definido en clases), indique el resultado considerando scope estático y luego scope dinámico. Recuerde que la diferencia entre scope estático y dinámico es el ambiente con el cual evaluamos el cuerpo de la función.

```
1)
               (run '{with {a 1}
                        \{\text{with } \{f \{\text{fun } \{x\} \{+ x a\}\}\}\}
                            {with {a 999}
                                {f 1}}})
   2)
               (run '{with {y 5}
                        {with {f {fun {x} {+ x y}}}}
                            {+ \{with \{y \{+ 8 y\}\} y\}}
                                {f 10}}})
b) Ahora indique que pasaría en los siguientes casos si modificamos el intérprete
   de la siguiente forma:
           [(app f arg)
               (def (closureV parameter body env-closure) (interp f env))
               (def new-env (extend-env parameter (interp arg env-closure)
                    env-closure))
                (interp body new-env)]
    1)
                    (run '{with {f {fun {x} {+ x 1}}}
                                {with {a 999}
                                       {f a}})
   2)
                    (run '{with {a 1}
```

{with {f {fun {x} {+ x 1}}} {with {a 999}}

{f a}}})