**TC2006 – Lenguajes de Programación***Funciones de Orden Superior en Haskell*

Matrícula1: A00811755 Matrícula2: A01088686

1. Programar la función de orden superior sin recursión explícita **aplica\_fns** en Haskell que aplique una lista de procedimientos binarios a un par de valores. Probar con:

> aplica\_fns [(+),(-),(\*)] 6 3 => [9,3,18,2]

> aplica\_fns [(<),(>),(==),(/=)] 6 3 => [False,True,False,True]

aplica:: a -> a -> (a -> a -> b) -> b

aplica num1 num2 proc = (proc num1 num2)

aplica\_fns:: [(a -> a -> b)] -> a -> a -> [b]

aplica\_fns lista num1 num2 = map (aplica num1 num2) lista

1. Programar la función de orden superior sin recursión explícita **reduce** en Haskell que regrese el resultado de aplicar un procedimiento binario a todos los valores de una lista que cumplan con un predicado. Probar con:

> reduce (\*) even? [1,2,3,4,5] => 8

> reduce (+) odd? [1,2,3,4,5] => 9

reduce :: (Int -> Int -> Int) -> (Int -> Bool) -> [Int] -> Int

reduce op func lista = foldl1 op (filter func lista)

1. Programar la función sin recursión **secuencia** en Haskell que genere listas con los números del 1 al N, utilizando la función until. Probar con:

> secuencia 3 => [1,2,3]

> secuencia 5 => [1,2,3,4,5]

secuencia :: Int -> [Int]

secuencia 0 = []

secuencia num = until (\x -> last x >= num) (\y -> y++ [(last y + 1)]) [1]