FACULTAD DE INGENIERIA DEL EJERCITO
Universidad de la Defensa Nacional

Paradigmas de programación IV
*Práctica de **Haskell***

Guía de ejercicios

Parte UNO

1. Programar una función que compute el número de elementos en una lista dada.
2. Agregar a la función información sobre el tipo. Para probarlo será necesario volver a cargar el archivo en ghci.
3. Programar una función que compute la media de una lista. Es probable que necesite hacer uso de la función fromIntegral para castear tipo de datos enteros.
4. Programar una función que convierta una lista en un palindromo. Por ejemplo, dada la lista [1,2,3], la función debería retornar [1,2,3,3,2,1].
5. Escriba una función que determine si la lista, pasada como parámetro, es un palíndromo.
6. Crear una función que ordena una lista de listas, basado en la longitud de cada sublista
7. Definir una función que une una lista de listas, usando a valor dado, como separador.
8. Definir el TDA árbol binario, y escribir una función que determine la altura del árbol.
9. Considere tres puntos (de dos dimensiones), a, b, y c. Si observamos el ángulo formado por la recta definida a través del segmento a-b y el segmento b-c, este resulta dar un giro hacia la izquiera, derecha, o no dar un giro y formar una recta. Definir un tipo de dato algebraico Direction que permita representar estas posibilidades.
10. Programar una función que calcule el giro generado por tres puntos bidimensionales, y que devuelva el tipo de dato algebraico definido en el punto anterior.
11. Definir una función que tome una lista de puntos bidimensionales y que compute la dirección de cada conjunto de tres puntos consecutivos. Dada una lista de puntos [a,b,c,d,e], debería comenzar computando el giro dado por [a,b,c], después por [b,c,d], y por último [c,d,e]. La función debería devolver una lista de Direction.

Paradigmas de programación IV
*Práctica de **Haskell***

Parte DOS

1. Escribir las funciones seguras de las estándares. Asegurarse de que no fallen para ningún caso. Es importante considerar el uso de los siguientes tipos:
    a. safeHead :: [a] -> Maybe a
    b. safeTail :: [a] -> Maybe [a]
    c. safeLast :: [a] -> Maybe a
    d. safeInit :: [a] -> Maybe [a]
2. Escribir la función *splitWith* que actúa de forma similar a *words,* pero que toma como parámetro un predicado y una lista de cualquier tipo, y luego splitea la lista en cada elemento para el cual el predicado es falso:
    a. splitWith :: (a -> Bool) -> [a] -> [[a]]
3. Usando el framewrok de comandos, escribir un programa que imprima por pantalla la primera palabra de cada línea de su input.
4. Escribir un programa que traspone el texto de un archivo. Por ejemplo:
    a. "hello\nworld\n" to "hw\neo\nlr\nll\nod\n".