PARADIGMAS DE PROGRAMACION

Trabajo Práctico N° 2 Fecha: 04/09/24

Tema: Programación Funcional. Lenguaje Haskell.

1. Explique qué hace el siguiente código Haskell y de un nombre más adecuado a la función.

```
misterio  x
   |  x <= 0   =  0
   |  x == 1   =  1
   | otherwise =  2*x + misterio (x - 1) - 1
```

2. Implemente en Haskell las siguientes funciones:

a. **divisiónEntera**: recibe dos números naturales y realiza la división entera entre dichos números mediante restas sucesivas.

b. **contarMayores**: recibe una lista de números enteros y un número X, y cuenta cuantos elementos de la lista son mayores que X.
   i. Realice una versión con *guards*
   ii. Realice una versión con *pattern matching*
   iii. Realice una versión con *list comprehension*

c. **subLista**: recibe una lista y un número natural n y retorna una lista con los primeros n elementos de la lista dada. No utilice *take*.

d. **sonIguales**: recibe dos listas y devuelve True si son iguales, caso contrario devuelve False.

e. **intersección**: recibe dos listas y devuelve una nueva lista con los elementos que tienen en común las listas dadas.

f. **esVocal**: recibe un carácter y retorna True si es una vocal, caso contrario retorna False.

g. **contarVocales**: recibe una lista de caracteres y devuelve la cantidad de vocales que contiene.

h. **transformar**: recibe como parámetros una función **f** (de un argumento) y una lista y devuelve como resultado la lista recibida en la que cada uno de sus elementos ha sido transformado con la función **f**. *Realice una versión con Pattern Matching*

i. **tablaDel5**: recibe un valor n y construye una lista que contiene los números de la tabla del 5 que se encuentran en el intervalo [0,n]. *Realice una versión con List Comprehension.*

j. **paresOrdenados**: construye una lista con todos los pares ordenados (a,b) posibles teniendo en cuenta que ***a*** corresponde a todos los números pares y ***b*** a los números impares entre 0 y 5 donde se cumple la condición que a < b. *Realice una versión con List Comprehension.*

k. **verificar:** recibe un predicado p (un predicado es una función que devuelve un valor booleano) y una lista de elementos xs y devuelve True si todos los elementos de la lista satisfacen el predicado, caso contrario devuelve False. *Realice una versión con Guards*

l. **combinar**: recibe una función **f** y dos listas y retorna una nueva lista que resulta de combinar las listas aplicando la función **f**. La función **f** debe recibir como parámetro un elemento de cada lista a combinar por vez. Utilice la siguiente definición de tipo para su función:

```
combinarCon :: (a -> b -> c) -> [a] -> [b] -> [c]
```

m. **filtrarLista**: recibe un predicado y una lista y devuelve la lista de elementos que satisfacen el predicado. La signatura de la función debería ser:

```
filtrarLista :: (a -> Bool)  -> [a] -> [a]
```

i. Realice una versión con *guards.*
ii. *Realice una versión con pattern matching.*
iii. Realice una versión con *list comprehension.*

3. Estudie cómo se aplica la función foldl en Haskell y escriba una versión propia.
.