PARADIGMAS DE PROGRAMACION

Trabajo Práctico N° 3 Fecha: 03/09/25

Tema: Programación Funcional. Lenguaje Haskell.

1. Explique qué tarea realiza cada una de las siguientes funciones misterio en código Haskell y de un nombre más adecuado a las mismas.

```
misterio1  n
   |  n == 0   =  1
   |  n == 1   =  3
   | otherwise =  9 * misterio1 (n - 2)
```

```
misterio2 0 = 1
misterio2 n = (mod n 10) * misterio2 (div n 10)
```

2. Implemente en Haskell las siguientes funciones:

a. **productoPorSumasSucesivas** que reciba dos números naturales y realice el producto entre dichos números mediante sumas sucesivas.

b. **sumaDigitosPares** que reciba un número positivo y devuelva la suma de los dígitos pares que contiene.

c. **contarMenores** que dada una lista de números enteros cuente cuantos elementos de la lista son menores que 10.
   i. Realice una versión con *Guards*
   ii. Realice una versión con *Pattern Matching*
   iii. Realice una versión con *List Comprehension*

d. **eliminarIgualesX** que reciba una lista y un número X y elimine de la lista todos los elementos iguales a X.
   i. Realice una versión con *Guards*
   ii. Realice una versión con *Pattern Matching*
   iii. Realice una versión con *List Comprehension*

e. **diferencia** que reciba dos listas y devuelva los elementos de la primer lista que no se encuentran en la segunda lista.

f. **subLista1** que reciba una lista y un número natural n y retorne una lista sin los primeros n elementos de la lista dada. No utilice drop.

g. **transformar** que reciba como parámetros una función **f** (de un argumento) y una lista y devuelva como resultado la lista recibida en la que cada uno de sus elementos haya sido transformado con la función **f**.

h. **tablaDePares** que, dado un número natural n, construya una lista que contenga los números pares que se encuentran en el intervalo [0,n]. Use *List Comprehension.*

i. **paresOrdenados** que construya una lista con todos los pares ordenados *(x,y)* posibles teniendo en cuenta que *x* corresponde a todos los números pares e *y* a los números impares entre 0 y 5 donde se cumple la condición que *x+y* es menor a un valor Z dado. Use *List Comprehension.*

j. **verificar** que 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 retorna False.
   i. Realice una versión con *Guards*
   ii. Realice una versión con *Pattern Matching*
   iii. Realice una versión con *List Comprehension*

k. **combinar** que reciba una función **f** y dos listas y retorne una nueva lista que resulta de la combinación 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]
```

   i. Realice una versión con *Guards*
   ii. Realice una versión con *Pattern Matching*
   iii. Realice una versión con *List Comprehension*

l. **filtrarLista** que reciba un predicado y una lista y luego regresa 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 *list comprehension*

m. **subLista2** que reciba una lista, un elemento x y un número natural n y retorne una lista sin los primeros n elementos de la lista dada que sean mayores que x. Utilice *Pattern Matching.*

n. **insertarLista**: función que recibe dos listas y un número natural n y retorna la primer lista a la que se le agregaron los elementos de la segunda lista a partir de la posición n. Utilice *Guards.*

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