Nombre y apellido: Aldana Ayelén Mercado
Carrera: Lic. Ciencias de la Computación

L.U. o D.N.I.: 613/19
Cant. de hojas: 2

Aprobada

Departamento de Computación – FCEyN – UBA

Taller de Álgebra I - Parcial

VERANO 2020 – *26 de febrero de 2020*

Aclaraciones

- El parcial se aprueba con tres ejercicios bien resueltos.
- Programe todas las funciones en lenguaje Haskell. El código debe ser autocontenido. Si utiliza funciones que no existen en Haskell, debe programarlas. Incluya la signatura de todas las funciones que escriba.
- No está permitido alterar los tipos de datos presentados en el enunciado, ni utilizar técnicas no vistas en clase para resolver los ejercicios.

Ejercicio 1

Dadas $f : \mathbb{Z} \times \mathbb{Z} \times \mathbb{Z} \to \mathbb{Z}$ definida como:

$$f(i,j,k) = \begin{cases} \frac{i}{3} & \text{si } i \text{ es múltiplo de 3} \\ j^2 + k & \text{en caso contrario} \end{cases}$$

Implementar en Haskell la función `f :: (Integer,Integer,Integer) -> Integer`.

Ejercicio 2

Escribir una función que tome como parámetro un entero $n \geq 1$ y que retorne el resultado de la siguiente sumatoria:

$$S(n) = \sum_{i=1}^{2n} (i-1)^2.$$

Ejercicio 3

Programe la función `maximos :: [Integer] -> [Integer] -> [Integer]` que recibe dos listas de enteros de igual longitud y devuelve una lista de esa misma longitud con el elemento más grande de cada posición.

Por ejemplo:

```
maximos [107, 33, 45] [28, 42, 37] ⇝ [107, 42, 45]
maximos [] [] ⇝ []
```

Ejercicio 4

Programe la función `sumatoriaImpares :: [Integer] -> Integer`, que dada una lista de enteros positivos, devuelve la suma de los números impares de la lista que se encuentran en posiciones impares.

Por ejemplo:

```
sumatoriaImpares [], ⇝ 0
sumatoriaImpares [1,3,6,4,7], ⇝ 8
sumatoriaImpares [2,3,4,8], ⇝ 0
sumatoriaImpares [1,3], ⇝ 1
```

Ejercicio 5

Programe la función `buscarPosicion :: Integer -> [Integer] -> Integer` que dado un entero y una lista de enteros devuelve la posición de la primera aparición del entero en el lista. Si no aparece el entero en la lista, retornar 0.

Por ejemplo:

```
buscarPosicion 3 [(-8),3,5,(-9),3,4] ⇝ 2
buscarPosicion 1 [(-8),3,5,(-9),3,4] ⇝ 0
buscarPosicion 3 [] ⇝ 0
```

1) $f(i,j,K) = \begin{cases} \frac{i}{3} & \text{si } i \text{ es multiplo de 3} \\ j^2 + K & \text{en caso contrario} \end{cases}$

```
EsMultiploDe3 :: Integer -> Bool
EsMultiploDe3 i | mod i 3 == 0 = True
                | otherwise = False

f :: (Integer, Integer, Integer) -> Integer
f (i, j, K) | EsMultiploDe3 i == True = (div i 3)
            | otherwise = (j ^ 2 + K)
```

con poner EsMultiploDe3 i = mod i 3 == 0 ya estaba ✓

2) Escribir una función que tome como parámetro un entero $n \geq 1$ que retorne el resultado de la siguiente sumatoria

$$S(n) = \sum_{i=1}^{2n} (i-1)^2$$

```
sumatoriabase :: Integer -> Integer
sumatoriabase n | n == 1 = 0
                | n > 1 = (n-1) ^ 2 + sumatoriabase (n-1)

sumatoriafinal :: Integer -> Integer
sumatoriafinal n = sumatoriabase (2*n)
```

✓

3) Programe la función maximos :: [Integer] -> [Integer] -> [Integer] que recibe dos listas de enteros de igual longitud y devuelve una lista de esa misma longitud con el elemento más grande de cada posición

```
maximos :: [Integer] -> [Integer] ->[Integer]
maximos [] [] = []
maximos (x:xs) (y:ys) | x>=y = [x] ++ maximos xs ys
                      | x<y = [y] ++ maximos xs ys
```

✓

4) Programe la funcion sumatoriaImpares :: [Integer] -> Integer que dada una lista de enteros positivos devuelve la suma de los numeros impares de la lista que se encuentran en posiciones impares.

```
esImpar :: Integer ->Bool

esImpar n | mod n 2 == 1 = True
          | otherwise = False


posicionesImpares [] = []
posicionesImpares [x] = [x]
posicionesImpares [x,y] = [x]
posicionesImpares (x:y:xs) = [x] ++ posicionesImpares xs
sumatoriaImpares :: [Integer] -> Integer
sumatoriaImpares [] = 0
sumatoriaImpares [x] | esImpar x == True = x
                     | otherwise = 0
sumatoriaImpares [x,y] | esImpar x == True = x
                       | otherwise = 0
sumatoriaImpares (x:y:xs) | esImpar x == True = x + sumatoriaImpares xs
                          | otherwise = 0 + sumatoriaImpares xs
```

✓

5) ~~buscarPosicion~~

Programe la función buscarPosicion :: Integer -> [Integer] -> Integer que dado un entero y una lista de enteros devuelve la posición de la primera aparición del entero en la lista. Si no aparece el entero en la lista retornar 0.

```
pertenece :: Eq a => a -> [a] -> Bool
pertenece _ [] = False
pertenece n (x:xs) = (x == n) || (pertenece n xs)
```

```
buscarPosición :: Integer -> [Integer] -> Integer
buscarPosición n [] = 0
buscarPosición n (x:xs) | pertenece n (x:xs) == False = 0
                        |(pertenece n (x:xs) == True) && (n == x) = 1
                        |(pertenece n (x:xs) == True) && (n /= x) = 1 + buscarPosicion n xs
```

✓