UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

Facultad de Ciencias

Estructuras Discretas
2025-2

Práctica 2

Profesor:
Ulises Rodríguez Domínguez

Ayudantes de teoría:
Irvin Javier Cruz Gónzalez

Ayudante de laboratorio:
Martínez Dámaso Raúl Eduardo

Objetivo y descripción.

El objetivo de esta práctica es que el alumnado programe su primer estructura de datos en Haskell y además note que temas como **definiciones recursivas** y **recursión** son vitales para poder lograrlo.

La primer estructura de datos que programará el alumnado será los **Árboles binarios**. Donde un Árbol Binario se define recursivamente de la siguiente manera:

1. ArbolBinarioVacio => Representa un árbol binario vacío.
2. Raiz a (ÁrbolBinario a) (ÁrbolBinario a) => Donde **Raiz a** representa la raíz de un árbol binario, **ÁrbolBinario a) (ÁrbolBinario a)** representa el árbol binario izquierdo y derecho respectivamente que son los hijos de la raíz mencionado al inicio.

En la sección **Ejercicios** se explicará los ejercicios que tendrá que programar el alumnado para simular el comportamiento de un Árbol Binario.

Ejercicios

1. **Longitud de un Árbol Binario.**

   Programa la función **longitud** que devuelva un entero que represente cuantas raíces tiene un Árbol Binario.

   ```
   ghci > longitud (Raiz 1 (Raiz 2 (ArbolBinarioVacio) (
   ArbolBinarioVacio)) (Raiz 3 (ArbolBinarioVacio) (ArbolBinarioVacio)))
   3
   ```

2. **Profundidad de un Árbol Binario.**

   Programa la función **profundidad** que calcule cuántos niveles tiene un Árbol Binario. Donde el nivel de un Árbol Binario vacío es de 0.

   ```
   ghci > profundidad (Raiz 1 (Raiz 2 (ArbolBinarioVacio) (
   ArbolBinarioVacio)) (ArbolBinarioVacio)))
   3
   ```

   NOTA: Para esta función si se puede utilizar la función **max** de Haskell.

3. **Ancho de un Árbol Binario.**

   Programa la función **ancho** que calcule cuántass hojas tiene un Árbol Binario. Donde una hoja es una raíz que no tiene Árbol Binario derecho e izquierdo.

   ```
   ghci > longitud (Raiz 1 (Raiz 2 (ArbolBinarioVacio) (
   ArbolBinarioVacio)) (Raiz 3 (ArbolBinarioVacio) (ArbolBinarioVacio)))
   2
   ```

4. **Tipos de recorridos de un Árbol Binario**:

Programa la función **recorrido** que devuelva una lista con los elementos de un árbol de acuerdo a alguno de los tres recorridos explicados a continuación:

(a) **Inorden**: Se visitan todas las raíces del árbol izquierdo con el mismo recorrido, después se visita la raíz principal y por último se visitan todas las raíces del árbol derecho con el mismo recorrido.

(b) **Preorden**: Se visita primero la raíz principal, después se visitan todas las raíces del árbol izquierdo con el mismo recorrido y por último se visitan todas las raíces del árbol derecho con el mismo recorrido.

(c) **Postorden**: Se visitan todas las raíces del árbol izquierdo con el mismo recorrido, después se visitan todas las raíces del árbol derecho con el mismo recorrido y por último se visita la raíz principal.

```
ghci > recorrido InOrden (Raiz 1 (Raiz 2 (ArbolBinarioVacio) (
ArbolBinarioVacio)) (Raiz 3 (ArbolBinarioVacio) (ArbolBinarioVacio)))
[2, 1, 3]
```

```
ghci > recorrido PreOrden (Raiz 1 (Raiz 2 (ArbolBinarioVacio) (
ArbolBinarioVacio)) (Raiz 3 (ArbolBinarioVacio) (ArbolBinarioVacio)))
[1, 2, 3]
```

```
ghci > recorrido PostOrden (Raiz 1 (Raiz 2 (ArbolBinarioVacio) (
ArbolBinarioVacio)) (Raiz 3 (ArbolBinarioVacio) (ArbolBinarioVacio)))
[2, 3, 1]
```

5. **Elementos de cada nivel en un Árbol Binario**:

Programa la función **niveles** que devuelva una lista de lista, donde cada sublista tiene todos los elementos si es que los hay del nivel n donde n $\in$ { 1...n}.

```
ghci > niveles (Raiz 1 (Raiz 2 (ArbolBinarioVacio) (
ArbolBinarioVacio)) (Raiz 3 (ArbolBinarioVacio) (ArbolBinarioVacio)))
[[1], [2, 3]]
```

Punto Extra: +2 => Programar la función usando una sola función auxiliar.

6. **Máximo o minimo de un Árbol Binario**:

Programa la función **maximo** ó **minimo** que devuelva el elemento máximo o mínimo de un Árbol Binario.

```
ghci > maximo (Raiz 1 (Raiz 10 (ArbolBinarioVacio) (
ArbolBinarioVacio)) (Raiz 4 (ArbolBinarioVacio) (ArbolBinarioVacio)))
10
```

Punto Extra: +.5 => Programar las dos funciones.

7. **Eliminar un elemento en un Árbol Binario ordenado.**:

Para este ejercicio tendrán que asumir que el Árbol Binario que se le pasará a la función está ordenado, es decir, cumple lo siguiente:

- Todos los elementos que estén a la izquierda de una raíz en específico son menores a este.
- Todos los elementos que estén a la derecha de una raíz en específico son mayores a este.

Programa la función **eliminar** que elimina un elemento en específico en un Árbol Binario ordenado y devuelve de igual manera un Árbol Binario ordenado.

Ve el siguiente video que explica el procedimiento que se tiene que seguir para eliminar un elemento en un Árbol Binario ordenado:

`https://youtu.be/BgFWkvauQ4w?si=psvABrapGyZgOdkn`

```
ghci > eliminar (Raiz 5 (Raiz 4 (ArbolBinarioVacio) (
ArbolBinarioVacio)) (Raiz 10 (ArbolBinarioVacio) (ArbolBinarioVacio))
) 4

Raiz 5 (ArbolBinarioVacio) (Raiz 10 (ArbolBinarioVacio) (
ArbolBinarioVacio)))
```

Restricciones

1. Como hemos visto en clase poner casos base de más provoca que se hagan líneas de código innecesarias, por lo que se evaluará esto de acuerdo al criterio del ayudante.
2. Durante la clase se vieron las buenas prácticas que debe de tener su código de acuerdo a lo visto en la práctica 1, si vuelves a cometer estos errores se les restará medio punto en el ejercicio donde cometan este error.
3. Si se cambian las firmas de las funciones, es decir, ya sea el nombre o los tipos que se describen se les restará medio punto de cada función donde hayan hecho esto.

Entrega

1. Se entregará la url de su repositorio público en donde han estado subiendo sus prácticas.
2. La fecha de entrega será el día **16 de mayo del presente año a las 23:59 pm**.
3. Las prácticas son individuales.
4. Queda estrictamente prohibido que tengan dos archivos referenciando a la misma práctica, ya que pueden llegar a provocar confusiones. Si esto llegase a suceder se restará un punto a su calificación total.