Departamento de Computación
FCEFQyN, Universidad Nacional de Río Cuarto
Asignatura: Estructuras de Datos y Algoritmos - Algoritmos y Estructuras de Datos II
Primer Cuatrimestre de 2025

Práctica No. 6

Acceder al código de base de los ejercicios de esta práctica aceptando el siguiente assignment: `https://classroom.github.com/a/bTOjhcmF`.

1. Definir en Haskell el predicado *bisiesto* : $Nat \mapsto Bool$, que determina si un año es bisiesto. Recordar que los años bisiestos son aquellos que son divisibles por 4 pero no por 100, a menos que también lo sean por 400. Por ejemplo, 1900 no es bisiesto, pero 2000 si lo es.

2. Acceder al proyecto en el repositorio de la práctica `adtSet`, que incluye la clase `Set` (*SetADT.hs*) y una implementación parcial utilizando `listas` (*SetList.hs*). Completar el TAD con las operaciones: `insert`, `contains`, `union`, `difference` e `intersection`.

3. Siguiendo el ejemplo del ejercicio anterior, cree un proyecto `adtStack` en el repositorio de la práctica, que incluya la clase `Stack` (*StackADT.hs*) y una implementación parcial utilizando `listas` (*StackList.hs*) y completar el TAD con las operaciones: `pop`, `isEmpty` y `top`.

4. Defina en Haskell el tipo `FQuadratic`, de las funciones cuadráticas (polinomios de grado 2). Defina además las siguientes funciones:

   - *eval* :: $FQuadratic \mapsto Float \mapsto Float$, que evalúa una función cua-drática en un número real dado,
   - *root* :: $FQuadratic \mapsto$ *(Float,Float)*, que retorna las raíces de una función cuadrática.

5. Defina en Haskell el tipo *Nat*, de los números naturales, y las siguientes funciones:

   - *suma* :: $Nat \mapsto Nat \mapsto Nat$, que suma dos números naturales dados,
   - *NatToInt* :: $Nat \mapsto Integer$, que retorna el número entero correspondiente al número natural dado como parámetro,
   - *producto* :: $Nat \mapsto Nat \mapsto Nat$, que retorna el producto de dos número naturales dados.

   Instancie además el tipo *Nat* para las clases *Eq* y *Show*.

6. Defina en Haskell el tipo *BinTree a*, de los árboles binarios (genéricos) de elementos de tipo *a*, y las siguientes funciones:

   - *altura* :: $BinTree\ a \mapsto Integer$, que retorna la profundidad de un árbol binario,
   - *contains* :: $BinTree\ a \mapsto a \mapsto Bool$ retorna verdadero si esta el elem buscado en el árbol.
   - *cantidad_nodos* :: $BinTree\ a \mapsto Integer$, que retorna la cantidad de nodos de un árbol binario.
   - *inorder*, *postorder*, *preorder* :: $BinTree\ a \mapsto [a]$, que retorna, dado un árbol binario, la lista de sus elementos según los recorridos inorder, preorder y postorder.
   - *minElem* :: $Ord\ a \Rightarrow BinTree\ a \mapsto a$, que retorna el elemento más pequeño del árbol

   Instancie además el tipo $BinTree\ a$ para la clase *Eq* y para la clase *Show*.