Estructuras de Datos – 1er semestre 2020
Tecnicatura en Programación Informática
Licenciatura en Informática con Orientación en Desarrollo de Software
Universidad Nacional de Quilmes

Primer parcial

Ejercicio 1. El siguiente tipo algebraico sirve para representar el sistema de archivos de una computadora:

```
type Nombre = String
type Fecha = Int
data Sistema = Archivo Nombre Fecha | Carpeta Nombre Sistema Sistema
type Ruta = [Nombre]
```

Cada *archivo* tiene un nombre y una fecha de creación. Cada *carpeta* tiene un nombre y almacena en su interior (recursivamente) dos sistemas de archivos. Una *ruta* es una lista de nombres de carpetas/archivos. Por ejemplo, el siguiente es un sistema de archivos:

```
Carpeta "Documentos" (Carpeta "Fotos" (Archivo "foto1.jpg" 12) (Archivo "foto2.jpg" 9))
                     (Carpeta "Música" (Archivo "tema1.mp3" 41) (Archivo "tema2.mp3" 44))
```

La ruta `["Documentos", "Música", "tema1.mp3"]` identifica un archivo válido dentro de dicho sistema, que fue creado en la fecha `41`. Definir la función:

```
filtrarPorFecha :: [Ruta] -> Fecha -> Sistema -> [Ruta]
```

que recibe una lista de rutas, una fecha y un sistema de archivos, y devuelve la lista de rutas que representan los archivos **válidos** de dicho sistema que fueron creados en la fecha indicada. Notar por ejemplo que, en el sistema de arriba, `["Documentos"]` no es un archivo válido porque no identifica un archivo sino una carpeta, `["foto1.jpg"]` no es un archivo válido porque la ruta está incompleta y `["Documentos", "Fotos", "foto3.jpg"]` no es un archivo válido porque refiere a un archivo inexistente.

Ejercicio 2. El tipo abstracto de datos `Alcancia` cuenta con la siguiente interfaz:

- `nuevaA :: Alcancia` — crea una alcancía vacía.
- `ponerA :: Int -> Alcancia -> Alcancia` — agrega una moneda en la alcancía, con el valor indicado. Los valores posibles de las monedas son $1, $2 y $5.
- `contarA :: Int -> Alcancia -> Int` — devuelve la cantidad de monedas del valor indicado que contiene la alcancía. Por ejemplo `(contarA 5 (ponerA 5 (ponerA 1 (ponerA 5 nuevaA)))) = 2`.

Se pide definir la función:

```
vaquita :: [Alcancia] -> Alcancia
```

que devuelve una alcancía que junta todas las monedas que hay en todas las alcancías de la lista.

Ejercicio 3. Por razones estadísticas y de seguridad, un museo mantiene registro de las personas que ingresan y egresan del establecimiento. El tipo abstracto `Museo` cuenta con la siguiente interfaz, donde `type DNI = Int`:

- `nuevoM :: Museo` — crea un museo vacío.
- `entrarM :: DNI -> Int -> Museo -> Museo` — registra el ingreso de una persona, indicando su edad.
  Precondición: la persona no debe estar en el museo.
- `salirM :: DNI -> Museo -> Museo` — registra el egreso de una persona.
  Precondición: la persona debe estar en el museo.
- `cuantosDe :: Int -> Museo -> Int` — dada una edad, indica cuántas personas de esa edad se encuentran actualmente en el museo. Por ejemplo, `cuantosDe 21 museo` indica cuántas personas de exactamente 21 años se encuentran actualmente en el museo.

El museo se representa con la siguiente estructura:

```
data Museo = M (Map DNI Int)   -- A cada persona en el museo le asocia su edad.
               (MultiSet Int)  -- Multiconjunto de las edades de las personas
                               -- del museo, consideradas con su repetición.
```

Escribir la implementación completa de las funciones `nuevoM`, `entrarM`, `salirM` y `cuantosDe`. Indicar la complejidad temporal en peor caso de cada operación (usando la notación "$O(...)$"), suponiendo que tanto el `Map` como el `MultiSet` se representan sobre un árbol binario de búsqueda (BST) con algún invariante de balanceo (por ejemplo, AVL).

Ejercicio 4. El tipo abstracto de datos `Grilla` representa un tablero cuadrado de $n \times n$ celdas. Cada celda almacena un número (no negativo: $0, 1, 2, \ldots$). Tiene la siguiente interfaz:

- `nuevaG :: Int -> Grilla` — recibe un número n y crea una grilla de $n \times n$ con 0s en todas las posiciones.
- `ponerG :: Int -> Int -> Grilla` — recibe dos números i, j, ambos entre 0 y $n - 1$ inclusive, e incrementa en 1 el número que hay en la celda (i, j).
- `maxG :: Grilla -> (Int, Int)` — devuelve la coordenada que tiene el número más grande de toda la grilla. En caso de empate, puede devolver cualquiera de ellas.

Elegir una estructura para representar el tipo `Grilla` e implementar las funciones `nuevaG`, `ponerG` y `maxG`. Debe cumplir con los siguientes requisitos de eficiencia en peor caso:

- `ponerG` debe ser $O(\log n)$
- `maxG` debe ser $O(1)$

Ejercicio 5. Un diario de viaje representa el recorrido de una persona a lo largo del tiempo. La persona comienza en una ciudad de origen y cada vez que se desplaza registra la fecha en que viajó y la ciudad de destino a la que llegó. La información se representa con los siguientes tipos de datos:

```
type Fecha = Int
type Ciudad = String
data DiarioDeViaje = EmpezarEn Ciudad | AgregarViaje DiarioDeViaje Fecha Ciudad
```

Por ejemplo `AgregarViaje (AgregarViaje (EmpezarEn "Buenos Aires") 2 "Bahía Blanca") 10 "Viedma"` es el diario de viaje de una persona que empezó en la ciudad de Buenos Aires, en el día 2 viajó a Bahía Blanca—donde permaneció 8 días—hasta el día 10, en el que viajó a Viedma. Suponemos que las fechas en el diario de viaje siempre aparecen en orden creciente.

El detective tiene una lista de personas con sus respectivos diarios de viaje, y conoce la fecha y el lugar del crimen. Definir la función:

```
contarSospechosos :: Map DNI DiarioDeViaje -> Fecha -> Ciudad -> Int
```

que dada una tabla (o sea, un `Map`) que indica el diario de viaje de cada una de las personas en observación—identificadas por su DNI, con `type DNI = Int`—y dadas además la fecha y la ciudad del crimen, devuelve la cantidad de sospechosos, es decir, la cantidad de personas que se encontraban en la ciudad del crimen en la fecha indicada.