Laboratorio de Lenguajes de Programación

USB / CI-3661 / Sep-Dic 2016

(Programación Orientada a Objetos – 15 puntos)

Déjà vu plegable

Árboles y arreglos plegables (3 puntos)

Considere la reapertura de la clase `Array` nativa de Ruby

```ruby
class Array
  def foldr e, b
    # Su código aquí
  end
end
```

En la cual el propósito de `foldr` es recibir un caso base y un *bloque* que será utilizado para recorrer la lista haciendo un pliegue de derecha a izquierda aplicando el *bloque* recibido.

Esto es, una llamada de la forma

```ruby
[1,2,3].foldr(z, f)
```

debe comportarse como si se evaluara `f(1, f(2, f(3, z)))`.

Ahora considere la siguiente definición para una clase que representa árboles multicamino (*Rose Trees*)

```ruby
class Rose
  attr_accessor :elem, :children
  def initialize elem, children = []
    @elem = elem
    @children = children
  end
```

```
  def add elem
    @children.push elem
    self
  end

  def foldr e, b
    # Su código aquí
  end
end
```

En la cual el propósito de `foldr` es recibir un caso base y un *bloque* que será utilizado para recorrer el árbol haciendo un pliegue de derecha a izquierda aplicando el *bloque* recibido.

Un ejemplo de un árbol

```
Rose.new(1, [
  Rose.new(2, [ Rose.new(4), Rose.new(5) ]),
  Rose.new(3, [ Rose.new(6) ])
])
```

Plegar es un comportamiento (3 puntos)

Se desea que usted implante una infraestructura de extensión del comportamiento de pliegue con la técinca de *mixins* estudiada en clase. Se espera que su *mixin* ofreza la siguiente interfaz de programación

- `null` que indica si la estructura de datos está vacía (`true`) o contiene elementos (`false`).
- `foldr1 &block` que recorre la estructura de manera similar a `foldr`, pero usando el primer elemento como el elemento neutro y hace el recorrido sobre el resto de la estructura. Debe lanzar (`raise`) una excepción si la estructura está vacía.
- `length` que indica el tamaño de la estructura de datos, o, mejor dicho, la cantidad de elementos que contiene.
- `all &block` que indica si **todos** los elementos de una estructura cumplen con un predicado.
- `any &block` que indica si **alguno (1 o más)** de los elementos de una estructura cumplen con un predicado.
- `to_arr` que construye una *nueva* instancia de la clase `Array` con todos los elementos contenidos, en orden de izquierda a derecha.

- **`elem to_find`** que indica si un elemento se encuentra en la estructura, comparando elementos con `==`.

Para la implantación de su *mixin* sólo puede suponer que las clases usuarias disponen del método **`foldr`**, y que para definir un método del *mixin* puede usar **`foldr`** y cualquier otro método previamente definido en el *mixin*.

Este ejercicio está inspirado en la clase de Haskell estudiada en clase, Data.Foldable[1], por lo que puede leer su documentación como orientación también.

Ejemplos

Los siguientes ejemplos ilustran el comportamiento esperado:

```
> [].foldr(0) {|x,s| x + s}
=> 0
> [1,2,3].foldr(0) {|x,s| x + s}
=> 6

> [].null
=> true
> [1,2,3].null
=> false

# maximum
> [].foldr1 {|x,y| if x>y then x else y end}
RuntimeError: foldr1: empty structure
> [1,5,3].fodlr1 {|x,y| if x>y then x else y end}
5

> t = Rose.new(0, [ Rose.new(2, [ Rose.new(4), Rose.new(6) ]) ])
=> #<Rose:0x00...>
> t.length
=> 4

> t.elem 6
=> true
> [1,2,3].elem 6
=> false

> t.to_arr
=> [0, 2, 4, 6]
> [1,2,3].to_arr
=> [1, 2, 3]
```

```
# all are even
> t.all {|x| x % 2 == 0}
=> true
> [1,2,3].all {|x| x % 2 == 0}
=> false

# any is odd
> t.any {|x| x % 2 != 0}
=> false
> [1,2,3].any {|x| x % 2 != 0}
=> true
```

Plegar el promedio (1 punto)

Finalmente, escriba un método para calcular el promedio de los valores en un árbol multicamino usando únicamente funciones del *mixin*. Debe escribir la función de manera tal que recorra el árbol **una sola vez** para los cálculos necesarios.

```
class Rose
  def avg
    # Su código aquí
  end
end
```

Para usarlo de la siguiente manera

```
> t = Rose.new(1, [ Rose.new(5, [ Rose.new(10), Rose.new(6) ]) ])
=> #<Rose:0x00...>
> t.avg
5.5
```

Detalles de la Entrega

La entrega se hará en un archivo `pR-<G>.tar.gz`, donde `<G>` debe ser sustituido por el número de grupo que le fue asignado. El archivo *debe* estar en formato TAR comprimido con GZIP – ignoraré, sin derecho a pataleo, cualquier otro formato que yo puedo descomprimir pero que *no quiero* recibir.

Ese archivo, al expandirlo, debe producir un *directorio* que *sólo* contenga:

- El archivo `foldable.rb` con la solución a la sección ***Déjà vu* plegable**

- El archivo `algebra.rb` con la solución a la sección **Algebra**

El proyecto debe ser entregado por correo electrónico a la dirección de contacto del profesor asignado, a más tardar el domingo 2016-11-27 a las 23:59. Cada minuto de retraso en la entrega le restará un (1) punto de la calificación final.

Referencias

1: Data.Foldable en Hackage