Programación declarativa 2018-2
Tarea 4

Fecha de entrega: 18 de mayo

Parte teórica (*si redactas por completo tu tarea en* LaTeX *tienes un punto extra*)

1. Sea $\mathcal{C}_{\mathbb{N}}$ como sigue:
   - $Ob_{\mathcal{C}_{\mathbb{N}}} = \mathbb{N}$
   - Si $n, m \in \mathbb{N}$, hay una flecha $n \to m$ sii $n \leq m$

   Muestra que $\mathcal{C}_{\mathbb{N}}$ es una categoría. ¿Qué hace un funtor $F : \mathcal{C}_{\mathbb{N}} \to \mathcal{C}_{\mathbb{N}}$?

   En adelante, sólo nos referiremos a la categoría **Hask**.

2. Sea `a` un objeto fijo y sea:
   $$F\text{:}\mathbf{Hask}\to\mathbf{Hask}$$
   $$\texttt{b} \mapsto \texttt{a->b}$$
   Demuestra que F es un funtor.

3. Sean `a,b` dos objetos fijos. Demuestra que:
   $$F_1\texttt{(c) = (a,b)->c}$$
   $$F_2\texttt{(c) = a->(b->c)}$$
   son funtores.

4. Demuestra que existen $\eta_1 : F_1 \Rightarrow F_2$ y $\eta_2 : F_2 \Rightarrow F_1$ transformaciones naturales tales que $\eta_2 \circ \eta_1 = F_1$ y $\eta_1 \circ \eta_2 = F_2$.

5. Sea η:`[]`$\Rightarrow$`[]` como sigue:
   $$\eta_{\texttt{a}}\texttt{:[a]}\to\texttt{[a]}$$
   $$\texttt{xs} \mapsto \texttt{sort xs}$$
   donde *sort* es cualquier función que ordena una lista dada. ¿Es η una transformación natural? Justifica tu respuesta.

6. Demuestra que η : `Maybe` $\Rightarrow$ `[]` definida como:
   $$\eta_{\texttt{a}}\texttt{:Maybe a}\to\texttt{[a]}$$
   $$\textit{Nothing} \mapsto \texttt{[]}$$
   $$\textit{Just}\ \texttt{x} \mapsto \texttt{[x]}$$
   es una transformación natural.

7. Demuestra que `Maybe` **: Hask**$\to$**Hask** es un funtor. Además:
   - Demuestra que `Maybe` es una mónada (desde el punto de vista categórico).
   - Demuestra que `Maybe` es una terna de Kleisli.

Parte práctica

- Implementa la función `goldbach::Int->[(Int,Int,Int)]` que hace lo siguiente: dado n un entero, si $n > 5$, devuelve una lista con ternas (q_1, q_2, q_3) tales que $q_1 + q_2 + q_3 = n$ y cada q_j es un número primo; `[]` en otro caso.
  Debes implementarla con listas por comprensión, con notación *do* y únicamente con los operadores $\gg$,$\gg=$, *guard* y *return*.

- Haz que el tipo `Arreglo` visto en clase forme parte de la clase `Functor` y demuestra que cumple las propiedades funtoriales.
- Con la mónada de estado implementa las funciones:
  - `minArrState::Ord a=>Arreglo a->Int->Int`
  - `selectionSortState::Ord a=>Arreglo a->Arreglo a`

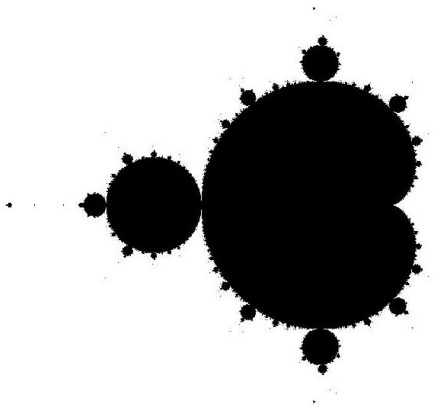
  Debes usar las funciones:
  - `getState = ST $ s -> (s,s)`
    Devuelve el estado actual.
  - `updState s = ST $ \_ -> ((),s)`
    Actualiza el estado.
- Utilizando la mónada de estado, escribe una función llamada `length1::[Int]->State Int Int` que, dada xs, devuelva (la longitud de xs, el número de veces que aparece 1 en xs). No puedes usar `getState` ni `updState`.
- Con la mónada de continuaciones implementa las funciones:
  - `lengthcps::[a]->Continuation r Int`
  - `takecps::Int->[a]->Continuation r [a]`
  - `dropcps::Int->[a]->Continuation r [a]`
- Modifica la función *dieta* vista en clase de tal manera que en lugar de recibir la comida de cada día de parte del usuario se le asigne aleatoriamente de una lista de veinte platillos diferentes.
- Modifica la función *getPass2* vista en clase de tal manera que si el usuario ingresa una contraseña incorrecta muestre el error del usuario en pantalla y vuelva a pedirle al usuario una contraseña hasta que lo haga de manera correcta.

Fractales

El conjunto de Mandelbrot es un subconjunto de $\mathbb{C}$. Para saber si un punto $c \in \mathbb{C}$ pertenece al conjunto de Mandelbrot se considera la siguiente sucesión:

$$S_c = \begin{cases} z_0 = 0 \\ \\ z_{n+1} = {z_n}^2 + c \end{cases}$$

De esta manera, $S_c = \{0, c, c^2 + c, (c^2 + c)^2 + c, \dots\}$. Si para todo $w \in S_c$ se cumple que $|w| \leq 2$, entonces c pertenece al conjunto de Mandelbrot.



Dado que S_c es infinito, nuestro criterio será revisar los primeros 200 elementos de S_c, y si ninguno de ellos tiene una norma mayor a 2, entonces c está en el conjunto de Mandelbrot. Si $w = a + bi$, la norma de w se calcula como $|w| = \sqrt{a^2 + b^2}$, por lo que, para nuestro problema, podemos decir que c está en el conjunto de Mandelbrot si para todo $w = a + bi \in S_c$ se tiene que $a^2 + b^2 \leq 4$.
Para esto, considera:

type $\mathbb{R}$ = Double
data $\mathbb{C}$ = Punto ($\mathbb{R}$,$\mathbb{R}$)

- Haz que el tipo $\mathbb{C}$ forme parte de la clase Num. Basta que definas la suma y el producto.
- Define la función norma::$\mathbb{C}$->$\mathbb{R}$ que calcule la norma de un número complejo sin usar raíz cuadrada como se mencionó arriba.
- Define la función plano::$\mathbb{R}$->[$\mathbb{C}$] que recibe un número n y devuelve una cuadrícula con n^2 números complejos dentro del cuadro de lado 2 con centro en el origen. Cuida que los puntos estén correctamente distribuidos para que tu fractal se visualice de forma correcta.
- Define la función mandelbrot_set::[$\mathbb{C}$]->String que recibe una lista de números complejos y devuelve una cadena como sigue: si c está en el conjunto de Mandelbrot, pinta "1 "; en otro caso, "0 ". Nota que es un 1 seguido de un espacio, o un 0 seguido de un espacio.
- Define la función mandelbrot_set_parallel::[$\mathbb{C}$]->String que hace lo mismo que la función anterior pero en paralelo.
Para pintar los fractales se utilizará el formato de imagen PPM (portable pixmap format) el cual es un archivo de texto plano que sigue el siguiente esquema:

P2
ancho largo
escala
la imagen como una cadena de texto

Por ejemplo:

```
P2
24 7
15
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  3  3  3  3  0  0  7  7  7  7  0  0 11 11 11 11  0  0 15 15 15 15  0
0  3  0  0  0  0  0  7  0  0  0  0  0 11  0  0  0  0  0 15  0  0 15  0
0  3  3  3  0  0  0  7  7  7  0  0  0 11 11 11  0  0  0 15 15 15 15  0
0  3  0  0  0  0  0  7  0  0  0  0  0 11  0  0  0  0  0 15  0  0  0  0
0  3  0  0  0  0  0  7  7  7  7  0  0 11 11 11 11  0  0 15  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
```

genera la imagen:

La cadena que representa la imagen puede ir en una sola línea. En nuestro caso sólo usaremos dos colores por lo que la escala será 1.

- Haz un programa que reciba como argumentos de la terminal dos cadenas *tam* y *mod* que indiquen el tamaño de fractal a generar y la forma de generarlo, "seq" para evaluación secuencial y "par" para hacerlo en paralelo. Tienes un punto extra si al hacerlo en paralelo se aprovechan todos los sparks generados.