Coursework 3

Lists

In this coursework we develop a simple program for the graphical display of regions in the 2-dimensional plane.

We represent a region as a *bitmap*, that is, a function $f : \mathbb{R} \times \mathbb{R} \to \mathbb{B}$ where $\mathbb{R}$ is the set of real numbers and $\mathbb{B} = \{T, F\}$ is the set of Booleans: $f(x, y) = T$ means the point $(x, y)$ is in the region represented by $f$, $f(x, y) = F$ means it is outside.

```haskell
type Pt = (Float, Float)
type Bitmap = Pt -> Bool

discBm :: Bitmap
discBm (x, y) = x^2 + y^2 <= 1
```

A *picture* is a region restricted to a rectangular segment of the plane, represented as a list of lists of Booleans.

```haskell
newtype Picture = Pic [[Bool]]

Example:

```haskell
discPic :: Picture
discPic =
  Pic [[False, False, False, False, False, True, False, False, False, False, False],
       [False, True, True, True, True, True, True, True, False, False, False],
       [False, True, True, True, True, True, True, True, True, False, False],
       [False, True, True, True, True, True, True, True, True, True, False],
       [True, True, True, True, True, True, True, True, True, True, True],
       [False, True, True, True, True, True, True, True, True, True, False],
       [False, True, True, True, True, True, True, True, True, True, False],
       [False, False, False, False, False, True, False, False, False, False, False]]
```
The picture discPic has been obtained by sampling the bitmap discBm at $11 \times 9$ equidistant points in the square $\{(x, y) \mid -1 \leq x, y, \leq 1\} \subseteq \mathbb{R} \times \mathbb{R}$. More precisely, the sampling points are of the form $(-1 + 0.2 * j, 1 - 0.25 * i)$ where $j \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$ and $i \in \{0, 1, 2, 3, 4, 5, 6, 7, 8\}$.

In general, the following additional data are required to compute a picture from a bitmap:

- A rectangle, given by its bottom left and top right corner (in our example $(-1, -1)$ and $(1, 1)$)
- Two positive integers, $n, m$, determining the numbers of sampling points inside the rectangle (in our example 10 and 8).

We call these data a *setting*:

```
type Setting = (Pt, Pt, Int, Int)
```

**Question 1.** Define a function `takePicture :: Setting -> Bitmap -> Picture` that computes a picture from a given setting and a bitmap. 

A simple way of displaying pictures is to transform them into strings as follows: A picture

```
Pic [bs1, bs2, ..., bsm]
```

is transformed into the string

```
('\n':s1) ++ ('\n':s2) ++ ... ++ ('\n':sm)
```

where '\n' is the *newline character* and each string $s_i$ is obtained from the Boolean list $b_i$ by replacing True with '@' and False with '.'. For example, the picture

```
Pic [[True, True, True], [True, True, False], [True, False, False]]
```

is transformed into the string "\n@@@\n@@.\n@..".

**Question 2.** Define a function `showPicture :: Picture -> String` that computes a string from a picture as described above. 

**Question 3.** Put the type `Picture` into the type class `Show` using the function `showPicture`. 

If Questions 1., 2. and 3. have been solved correctly, evaluation of the expressions `discPic` and `takePicture ((-1,-1),(1,1),10,8) discBm` should yield the same result, which should be displayed by Haskell as follows:

```
.....@.....
..@@@@@@@.. 
..@@@@@@@@@.
..@@@@@@@@@.
..@@@@@@@@@.
..@@@@@@@@@.
..@@@@@@@.. 
.....@.....
```
Question 4. Using the setting

\[ \text{set0 :: Setting} \]
\[ \text{set0} = ((-1.5,-1.5),(1.5,1.5),80,40) \]

(and a small font for displaying the results) take pictures of the following regions:

(a) \{ (x, y) \mid y \leq x \}.
(b) \{ (x, y) \mid x \cdot y > 0 \}.
(c) \{ (x, y) \mid x \cdot y > 0.25 \}.
(d) A square with vertices \((-1, -1), (1, -1), (1, 1), (-1, 1)\).

[20 marks]

Question 5. Define the following operations on pictures:

(a) Appending two pictures vertically (second below first).
(b) Appending two pictures horizontally (side by side).
(c) Flipping a picture upside down.
(d) Flipping left and right of a picture.
(e) Inverting a picture (complement, that is, pointwise negation).
(f) Combining two pictures disjunctively (union, pointwise \((\mid\mid)\)).
(g) Combining two pictures conjunctively (intersection, pointwise \((\&\&)\)).
(h) Combining two pictures exclusively (pointwise \((/=))\).

In questions (a), (b), (f), (g) and (h) it may be assumed that the given pictures are of the same size. Test the operations with the examples from question 3 (using a different setting if necessary).

[40 marks]
Question 6. The logistic map

\( \varphi : [0,1] \to [0,1], \quad \varphi x = 4 \times x \times (1 - x) \)

is a simple example of a dynamical system with chaotic behaviour. It is often used to test implementations of real number operations. \( ([0,1] = \{ x \in \mathbb{R} \mid 0 \leq x \leq 1 \}) \)

More precisely, for every starting point \( x \in [0,1] \) the orbit of the logistic map,

\( \{ \varphi^n x \mid n = 0, 1, 2, 3, \ldots \} = \{ x, \varphi x, \varphi(\varphi x), \varphi(\varphi(\varphi x)), \ldots \} \)

is a highly irregular set which, for large \( n \) is difficult to compute because of the accumulation of rounding errors.

Take pictures of the iterates, \( \varphi^n \), of the logistic map for \( n = 1, \ldots, 10 \), that is, take pictures of the regions

\( \{ (x, y) \mid x, y \in [0,1], y < \varphi^n x \} \)

Question 7. The Mandelbrot set is the set of all complex numbers \( c \in \mathbb{C} \) with the property that the mapping

\( f_c : \mathbb{C} \to \mathbb{C}, \quad f_c z = z^2 + c \)

has a bounded orbit at 0, that is the set

\( \{ |f^n_c| \mid n = 0, 1, 2, 3, \ldots \} \)

is bounded. The Mandelbrot set is a fractal, that is, a set with a highly complicated microstructure (see e.g. Wikipedia for more information and images).

By identifying a complex number \( c = x + iy \in \mathbb{C} \) with the point \( (x, y) \) we may view the Mandelbrot set as a region in the 2-dimensional plane.

Take a picture of the following region that approximates the Mandelbrot set:

\( \{ c \in \mathbb{C} \mid |f^n_c| < 2 \text{ for all } n \in \{0, 1, 2, \ldots, 20\} \} \)

Reminder on how to calculate with complex numbers:

\[
\begin{align*}
(x + y \ast i) + (x' + y' \ast i) &= (x + x') + (y + y') \ast i \\
(x + y \ast i) \ast (x' + y' \ast i) &= (x \ast x' - y \ast y') + (x \ast y' + y \ast x') \ast i \\
|x + y \ast i| &= \sqrt{x^2 + y^2}
\end{align*}
\]

Hence \( |x + y \ast i| < 2 \) if and only if \( x^2 + y^2 < 4 \).

CS-221 students submit Questions 1-5.
CS-M36 students submit Questions 1-6.
Questions 7 is voluntary.

Submission: Please follow the same rules as for Coursework 1. It is recommended to include the (commented out) testing results at the end of the coursework.

Due date (for CS-221 and CS-M36 students): Thursday, 23 November 2006.