Programming languages (TC-2006)

Homework 10

In this homework, you will practice with data structures implemented in the Haskell language. Please consider that the purpose of this homework is to allow you to practice and identify strengths and weaknesses. Then, implement these functions as requested and avoid using any built-in functions that already do what you are requested to implement. Since this homework focuses on user-defined data types, consider that, in all the functions requested, you must use implement and use the requested data type. Failing to do so will cancel any points related to such a function.

1 Distance between two points (10%)

1.1 `distance` (10%)

Write a function in Haskell that receives two Points and returns the Euclidean distance between the two points ($d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$), where the user-defined data type `Point` is defined as follows:

```
data Point = Point Double Double deriving Show
```

For example, calling your function with the arguments `Point 10 20` and `Point 5 15` must return 7.07106 because that is the Euclidean distance between these two points.

2 Complex numbers (40%)

A complex number is a number that can be expressed in the form $a + bi$, where a and b are real numbers and i is the imaginary unit, satisfying the equation $i^2 = -1$. The real number a is called **the real part** of the complex number $a + bi$; the real number b is called **the imaginary part** of $a + bi$. For this assignment, the `Complex` data type is defined as follows:

```
data Complex = Complex Int Int deriving Show
```

By using the previous definition, implement four functions to interact with such a data type.

2.1 `add` (10%)

Write a function in Haskell that takes two complex numbers and adds them. Complex numbers are added by separately adding the real and imaginary parts of the operands. This is: $(a+bi)+(c+di) = (a+c)+(b+d)i$.

2.2 `multiply` (10%)

Write a function in Haskell that takes a complex number and a `Int` and multiplies them. In this case, the result of $c \times (a + bi)$ is $(a \times c) + bi$.

2.3 `getReal` (10%)

Write a function in Haskell that returns the real part of the complex number.

2.4 `getImaginary` (10%)

Write a function in Haskell that returns the imaginary part of the complex number.

3 The RGB color mode (30%)

Define a new data type for representing colors in RGB color mode. The RGB color model is an additive color model in which red, green and blue light are added together to reproduce a broad array of colors. For this assignment, the `RGB` data type is defined as follows:

```
data RGB = RGB (Int, Int, Int) deriving Show
```

For example, color red would represented as `RGB (255, 0, 0)`, green as `RGB (0, 255, 0)` and blue as `RGB (0, 0, 255)`. Different combinations of the three values result in different colors. For example, orange would represented as `RGB (255, 165, 0)`. By using the previous definition, implement four functions to interact with such a data type.

3.1 `getR` (6%)

Write a function in Haskell that returns the red component of the color.

3.2 `getG` (6%)

Write a function in Haskell that returns the green component of the color.

3.3 `getB` (6%)

Write a function in Haskell that returns the blue component of the color.

3.4 `getMaxComponent` (6%)

Write a function in Haskell that returns the component with the highest value. For example, if a color is defined as `RGB (100, 30, 180)`, the function `getMaxComponent` must return `("Blue", 180)`. Please note that the result is a pair.

3.5 `combine` (6%)

Write a function in Haskell that receives two RGB colors and combines them by calculating the average of each of their respective components. For example, given the colors `RGB (100, 200, 180)` and `RGB (150, 140, 220)`, the result of combining those colors would be `RBG (125, 170, 200)`.

4 Playing cards (20%)

In card games such as Poker and Blackjack each one of the 52 cards have a suit and a rank (a number from 2 to 10 and four additional elements: Jack, Queen, King and Ace, which represent the numbers 11, 12, 13, and 14, respectively). There are 4 suits and 13 ranks in each suit. For this assignment, the user-defined data types to handle the cards are defined as follows:

```
data Suit = Diamonds | Clubs | Hearts | Spades deriving Show
data Color = Red | Black deriving Show
data Rank = Jack | Queen | King | Ace | Two | Three | Four | Five | Six |
    Seven | Eight | Nine | Ten deriving Show
data Card = Card Rank Suit deriving Show
```

Now that we have defined the `Card` data type, write two functions to interact with such a data type.

4.1 `getValue` (10%)

A function that returns the value of a card (the numeric value for its rank). For example, The value of `(Card (Queen, Hearts))` is 12 because the numeric value of the Queen is 12. Please note that the `Suit` is completely ignored for the calculation of the numeric value of the card.

4.2 `getColor` (10%)

A function that returns the color of a card (where `Color` is also a user-defined data type). For example, The color of `(Card (Queen, Hearts))` is `Red` because it is the color of any card in the suit `Hearts`. Please note that the `Rank` is completely ignored for the calculation of the color of the card.

Deliverables

Prepare an HS file that contains the functions requested (in its corresponding module) and submit it to Canvas. **Please, do not submit other formats but HS**. To prepare your HS file, use the code template distributed along with this document. The template contains some test cases for each function to help you verify that your codes work as requested.

I promise to apply my knowledge, strive for its development, and not use unauthorized or illegal means to complete this activity, following the Tecnológico de Monterrey Student Code of Honor.