MATHFUN
Discrete Mathematics and Functional Programming

Worksheet 1: Introduction to Functional Programming

Introduction

This worksheet is intended to get you started using the Haskell language. Begin by using a text editor (e.g. NotePad++) to produce a Haskell script (with suffix .hs) containing the function definitions from the lecture. Load this script into GHCi, and experiment by evaluating some expressions.

You can add the solutions to the programming exercises at the end of the script; each time you make a change to the script, load (or reload) it into GHCi. Make sure that you thoroughly test the solutions to each of the exercises.

Programming exercises

1. Write a function which multiplies its argument by 10:
   ```
   timesTen :: Int -> Int
   ```
   (For example, `timesTen 5` gives 50.)

2. Write a function which gives the sum of three integers:
   ```
   sumThree :: Int -> Int -> Int -> Int
   ```

3. Using the constant `pi` and the power operator `^`, write a function which gives the area of a circle given its radius:
   ```
   areaOfCircle :: Float -> Float
   ```

4. Using the definition of `areaOfCircle`, write a function that gives the volume of a cylinder given its length and cross-sectional radius:
   ```
   volumeOfCylinder :: Float -> Float -> Float
   ```
   (The volume is the cross-sectional area times the length.)

5. Write a function that takes four floats representing the coordinates x_1, y_1, x_2, y_2 of two points, and gives the distance between the points:
   ```
   distance :: Float -> Float -> Float -> Float -> Float
   ```
   Use the formula:
   $$distance = \sqrt{(y_1 - y_2)^2 + (x_1 - x_2)^2}.$$

6. Write a function which returns True if, and only if, all of its three arguments are all different from one another:
   ```
   threeDifferent :: Int -> Int -> Int -> Bool
   ```

7. Using the `mod` function, write a function that tests whether one integer is divisible by another:
   ```
   divisibleBy :: Int -> Int -> Bool
   ```
   (E.g., `divisibleBy 10 2` is True and `divisible 10 3` is False.)

8. Using the definition of `divisibleBy`, write a function which determines whether its argument is an even number:

   ```
   isEven :: Int -> Bool
   ```

9. Write a function:

   ```
   averageThree :: Int -> Int -> Int -> Float
   ```

   which gives the average of three integer values. Note that Haskell will treat the inputs as Ints, and the output will need to be a Float (since those are the types given in the type declaration), but it doesn't automatically convert from one type to the other. Use the `fromIntegral` function to do this.

10. Using a conditional expression, write a function that gives the absolute value of an integer (i.e. gives a non-negative value):

    ```
    absolute :: Int -> Int
    ```

    Note that if you write:

    ```
    f -3
    ```

    where `f` is any function in Haskell, it will be interpreted as `(f -) 3` and thus give an error – you'll see why in a later lecture. You may need to use your own parentheses in order to avoid such an error.