Homework 1

Functional Programming

due: 2024-02-07

Place your solutions in a module named `Homework1` in a file with path `homework/Homework1.idr` within your course git repository. Please submit *only* your Idris source file. Depending on your OS and program editor, you may need to modify the `.gitignore` file to exclude unwanted cruft.

At the beginning of the file include a comment containing your name. Precede each problem's solution with a comment specifying the problem number.

Whether or not it is complete, the solution file that you submit should load without errors. If you encounter a syntax or type error that you are unable to resolve, use comments or goals to isolate it from the part of the file that is interpreted by Idris.

Your solutions will be pulled automatically for marking shortly after the due date.

Problem 1
An important boolean function is `implies`, which takes two `Bool` inputs and returns `True` just in case the second is at least as true as the first:

↓ `implies` →	`True`	`False`
`True`	`True`	`False`
`False`	`True`	`True`

Write this function in Idris using definition by pattern matching.

Problem 2
Write a three-element type called `Roshambo` with elements called `Rock`, `Paper`, and `Scissors`.

Problem 3
Write a function representing the `beats` relation for the game *rock paper scissors*:

```
beats  :  Roshambo -> Roshambo -> Bool
```

For example:

```
Homework1> Paper `beats` Rock
True
Homework1> Paper `beats` Scissors
False
```

Problem 4
Redefine the `Shape` type from week 2, adding an additional element constructor called `RegularPolygon` that takes as arguments a `Nat` number of sides and a `Double` side length.

Note: In Euclidean geometry polygons should have at least 3 sides. We wll learn how to enforce this condition later, but for now 0-, 1-, or 2-gons are okay.

Problem 5
Write the `perimeter` function for `Shape`s:

```
perimeter  :  Shape -> Double
```

Note: For this problem do not use any casting of types (which we haven't discussed yet), instead think about how to solve it using methods that we have learned.

Problem 6
The *Fibonacci function* was discovered over 2200 years ago and describes a surprising array of phenomena. It has the following type:

```
fib  :  Nat -> Nat
```

and is recursively defined as:

$$\text{fib } n = \begin{cases} n & \text{if } n = 0 \text{ or } n = 1 \\ \text{fib } (n-1) + \text{fib } (n-2) & \text{otherwise} \end{cases}$$

Write the Fibonacci function in Idris using pattern matching on the argument `Nat`. Confirm that it returns correct results for some low argument values according to `https://oeis.org/A000045`.

Problem 7
Write a recursive definition for the exponentiation function on the natural numbers, m^n:

```
exp  :  Nat -> Nat -> Nat
```

For example:

```
Homework1> exp 2 0
1
Homework1> exp 2 1
2
Homework1> exp 2 2
4
Homework1> exp 2 3
8
```

Hint: Think about in which argument this function is recursive, and use your recursive definition of multiplication from lab 2 as a guide.