Homework 5

ITI0212

Due: 05/31/2024

Place your solutions in a module named `Homework5` in a file with path `homework/Homework5.idr` within a repository called `iti0212-2024` on the TalTech GitLab server (`https://gitlab.cs.ttu.ee/`). Submit only your Idris source file. Do not include any temporary files or build artifacts, such as `.swp`, `.tmp`, `.ttc`, or `.ttm` files.

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 holes 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.

All of your solutions in this assignment should be *total*. Idris can warn you of potential non-totality if you include the `%default total` directive at the beginning of your module. For this assignment we will not accept any solution that uses Idris's `rewrite` mechanism, which we did not introduce in this course. Instead, you should the user-definable and type-directed alternatives that we did learn about.

Problem 1. Convince Idris of the following mathematical fact:

```
plus_one_right  :  {n : Nat} -> n + 1 = S n
```

Problem 2. Write the following function that reverses the order of the elements in a vector:

```
vect_reverse  : {n : Nat} -> Vect n a -> Vect n a
```

For example:

```
Homework5> vect_reverse []
[]
Homework5> vect_reverse [1]
[1]
Homework5> vect_reverse [1, 2]
[2, 1]
Homework5> vect_reverse [1, 2, 3]
[3, 2, 1]
```

Hint: You can reverse the element order using the simple structurally recursive algorithm (for example in homework 4, problem 9), but you will find that in the recursive case you need to do something in order to get from the computed result type to the desired result type.

Problem 3. Prove that if two propositions are decidable then so is their conjunction:

```
dec_and  :  Dec p -> Dec q -> Dec (p `And` q)
```

Prove that if a proposition is decidable then so is its negation:

```
dec_not  :  Dec p -> Dec (Not p)
```

Problem 4. Write the function that returns half of a natural number under the *constraint* that the argument is *even*. For example:

```
Homework5> half 0
0 : Nat
Homework5> half 2
1 : Nat
Homework5> half 3
Error: Can't find an implementation for IsEven 3.
```

Hint: Recall that the error message, "can't find an implementation for ..." can mean "can't satisfy the constraint ..." because Idris uses the same *search* mechanism for interface resolution and constraint solving.

Problem 5. Suppose you are hired to work on a video game. The state of the player in the game is represented by the following record type:

```
record  PlayerState  where
constructor PS
health  :  Fin 11
wealth  :  Fin 101
```

Your job is to write the function that implements the interaction of the player hiring a healer in the game:

```
hire_healer  :  PlayerState -> PlayerState
```

In this interaction the player increases their health by one unit in exchange for 10 coins (the units of wealth). If the player does not have enough coins or already has maximum health then no transaction occurs and the player's state remains unchanged.

Note: Arithmetic in `Fin` types is *modular*; i.e., it "wraps around" in the same way as when we add or subtract time on a clock. But this need not concern us so long as we don't perform operations that overflow or underflow the `Fin` type.

Problem 6. Write a record type `Complex` for *complex numbers* with fields for the *real* and *imaginary* parts, both of type `Double`, and a constructor called `(!!)`. Don't forget to add an `infix` declaration for this operator. Then write a `Num` implementation for `Complex`. For example:

```
Homework5> 1 !! 2 + 3 !! 4
4.0 !! 6.0
Homework5> 1 !! 2 * 3 !! 4
-5.0 !! 10.0
Homework5> the Complex (fromInteger 7)
7.0 !! 0.0
```