Homework 6

Functional Programming

due: 2024-05-08

Place your solutions in a module named `Homework6` in a file with path `homework/Homework6.idr` within your course git repository. Please submit *only* your Idris source file and any Idris source files that it imports. At the beginning of the file include a comment containing your name. Precede each problem's solution with a comment specifying the problem number and `public export` each definition that you are asked to write so that it can be `import`ed for testing. All definitions in your file should be *total*, which you can ensure by using the `%default total` directive. The following questions refer to some of the definitions `export`ed from lectures 10, 11, and 12, so you should either `import` those modules or else copy the relevant definitions to your file. Recall that you may *not* use Idris's `rewrite` construct in any of your solutions.

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
Convince Idris that the sum of any number and itself is even:

```
double_even  :  (n : Nat) -> Even (n + n)
```

Hint: in the successor case it may help to find a proof of the following lemma:

```
    lemma  :  Even (S (S (n + n)))
```

and `transport` it in the indexed type family `Even` along an equality:

```
    path  :  S (S (n + n)) = (S n + S n)
```

Recall that you can write such local definitions using a `where` clause.

Problem 2
Convince Idris that the product of an even number and any number is even:

```
even_times_any  :  (m , n : Nat) -> Even m -> Even (m * n)
```

Hint: in the `SS_even` case it may help to find a proof of the following lemma:

```
    lemma  :  Even $ (n + n) + (m * n)
```

and `transport` it in the indexed type family `Even` along an equality:

```
    path  :  (n + n) + (m * n) = n + (n + (m * n))
```

Problem 3
Using the interpretations of `And`, `Or` and `Not` from lecture 12, convince Idris of the following *de Morgan laws*:

```
dm1  :  Not a `Or` Not b -> Not (a `And` b)
dm2  :  Not a `And` Not b -> Not (a `Or` b)
```

Problem 4
Using the interpretation of `Some` from lecture 12, convince Idris that every even number is the double of some other number:

```
evens_are_doubles  :  Even n -> Some Nat (\ m => m + m = n)
```