CS 457/557 Homework 7 – due 2pm, Tuesday, November 22,2005

Hand in your solutions on paper *and* email them to `cs457acc@cs.pdx.edu`. They should be placed in a single `hw8.hs` file, which should be an attachment.

1. Do Hudak exercise 13.3, with the following simplifications and guidelines. Define a function

```
clock :: Int -> Int -> Int -> Animation Graphic
```

that takes an initial time specified as three integers (hours, minutes, seconds) and returns an `Animation` rendering your clock. Then, write a function `main: IO ()` that obtains the current time (using the functions available in the `Time` library — Nov2002 Hugs users should get the bug-fixed version of `Time.hs` from the course web page) and calls `clock` with it.

Don't worry about making your clock too pretty, just so long as it is recognizable as a (non-digital!) clock. In particular, *don't* bother trying to label the face with numbers; this is surprisingly difficult to do given the framework Hudak has made available. You should try to have some kind of tick marks at the rim, however. Your clock "hands" can be very basic (e.g., just small circles). You'll find it convenient to make the clock face have radius of about 1.0 or 2.0 units (in the `Region` coordinate system). You'll need to recall some basic trigonometry, but the examples in the text should be enough of a reminder.

Finally, don't be distressed if Hugs can't run fast enough to keep your second hand moving smoothly.

2. Do Hudak exercises 14.6, 14.7, and 14.8, using the following modifications and guidelines. For 14.6, define a more general type of polynomials, namely

```
type Poly a = [a]
```

In addition to the four operators requested in the book, also define a function

```
toPoly :: [a] -> Poly a
```

that converts a (finite) list of coefficients into a polynomial by appending a stream of zeros.

Then do exercise 14.8. This will require you to define a distinct new type of polynomials, e.g.

```
newtype P a = P [a]
```

Then you should define instances

```
instance Num a => Num (P a) where ...
instance Fractional a => Fractional (P a) where ...
```

For the most part, you can fill out these instance declarations just by invoking the functions you wrote for 14.6. Don't worry about defining `abs` or `signum` methods for the `Num` instance (they don't make too much sense for polynomials). Also, although you will have to define `Polynomial` to be an instance of `Num`'s superclasses (`Eq` and `Show`), you *don't* actually have to define methods in these instances! (Remember that a missing method is allowed at compile time, although it will give an error at runtime if the method is invoked.)

Finally, do exercise 14.7 using the notation you've enabled in 14.8.