CSE 341 — Haskell Mini-Exercises # 3

These are questions for discussion in class. (You don't need to hand in anything.) The solutions are on the class web page.

1. What are the first ten elements in the following list?

   ```
   mystery = 1 : map (+2) mystery
   ```

2. The Prelude includes a function `repeat`. For example, `repeat 3` evaluates to the infinite list of threes, so that `take 5 (repeat 3)` evaluates to `[3,3,3,3,3]`. Write your own version, called `repeat'`. What is its type? (Write the most general possible type.)

3. Another useful function in the Prelude is `cycle`. For example, `cycle [1,2,3]` evaluates to the infinite list whose first 10 elements are `[1,2,3,1,2,3,1,2,3,1]`. Write your own version, called `cycle'`. What is its type? (Write the most general possible type.)

4. What happens when you apply `cycle` to an infinite list, for example `cycle [1..]`? (Hint: it is equivalent to another function in the Prelude. What is that function?)

5. Give a definition of a list `doubles` whose first element is 10, and whose n^{th} element is twice the $n-1^{st}$, i.e., `[10, 20, 40, 80, 160, 320, ....]`. To do this, write a recursive helper function `doubles_from` that takes a parameter `n` and returns a list of all the doubles starting at `n`.

6. Give another definition of `doubles` that looks like this: `doubles = 10 : something`. (So replace `something` with a suitable expression.) Don't use a helper function. Hint: this is using the prime-the-pump pattern for defining an infinite list that was also used in the definitions for the Fibonacci and Hamming numbers in the lecture notes.

7. Give yet another definition of `doubles` using the built-in function `iterate` from the Haskell prelude. This is defined as follows:

   ```
   iterate          :: (a -> a) -> a -> [a]
   iterate f x      =  x : iterate f (f x)
   ```

8. Define a Haskell list `dollars` that is the infinite list of amounts of money you have every year, assuming you start with $100 and get paid 5% interest, compounded yearly. (Ignore inflation, deflation, taxes, bailouts, the possibility of total economic collapse, and other such details.) So `dollars` should be equal to `[100.0, 105.0, 110.25, ...]`.

9. Define a variable `ints` whose value is the infinite list of all integers. It should be ordered in such a way that you can find any given integer after searching a finite number of elements in `ints`. In other words, this isn't going to work:

   ```
   ints = [1 ..] ++ [-1, -2 ..]
   ```