Recitation Session, November 8 2017

Please do not write on this sheet of paper
And do not use laptops during the session

For comprehensions and monads

Ex 1.1

Consider a directed graph given by its set of (directed) edges stored as a list of pairs of nodes:

```
type NodeId = Int
type DirectedEdge = (NodeId, NodeId)
type DirectedGraph = List[DirectedEdge]
```

Define, non-recursively, the `triangles` function that finds all cycles of length 3, with three distinct nodes, in the given graph. You should use a for comprehension.

```
def triangles(edges: DirectedGraph): List[(NodeId, NodeId, NodeId)] = for ...
```

Each cycle should appear only once. For instance, given the edges:

```
List((1, 2), (2, 3), (3, 1))
```

The should return **exactly one** of the three following possibilities:

```
(1, 2, 3), (2, 3, 1), (3, 1, 2).
```

You are free to decide which of the three you return.

Ex 1.2

After that, translate the `for` comprehension you wrote in the appropriate combination of `map/flatMap/filter` calls.

Ex 2.

We want to show that the `List` datatype is a monad, with `unit(x)` defined as `List(x)`. You should use inductive reasoning (when necessary) as well as the following axioms.

Axioms:

1. `Nil.flatMap(f) === Nil`
2. `(x :: xs).flatMap(f) === f(x) ++ xs.flatMap(f)`
3. `xs ++ Nil === xs`
4. `Nil ++ xs === xs`
5. `(xs ++ ys).flatMap(f) === xs.flatMap(f) ++ ys.flatMap(f)`
6. `(x :: xs) ++ ys === x :: (xs ++ ys)`
7. `List(x) === x :: Nil`
8. `(y => E)(x) === E'`
   `E'` is the result of carefully replacing y by the expression x in `E`.

Show the following monad laws:

1. Left unit law:
   `List(x).flatMap(f) === f(x)`
2. Right unit law:
   `list.flatMap(y => List(y)) === list`

3. (*Optional*) flatMap associativity:
   `list.flatMap(f).flatMap(g) === list.flatMap(y => f(y).flatMap(g))`

   Hint: Prove the property for Nil, List(x) and then for arbitrary lists by induction.

Optional exercise:

Prove the "axiom" number 5 using the other axioms.