Recitation Session Solutions, November 8 2017

Ex 1.1

```
// For equivalent cycles, we decide to return the cycle
// with the smallest element first.
def triangles(edges: DirectedGraph): List[(NodeId, NodeId, NodeId)] =
  for {
    e1 <- edges

    // The first node is the smallest of the cycle.
    if (e1._1 < e1._2)

    e2 <- edges

    // The two edges are connected and
    // there exists an edge between the two other end points.
    if (e1._2 == e2._1 && edges.contains((e2._2, e1._1)))

    // The first node is the smallest of the cycle.
    if (e1._1 < e2._2 && e2._1 != e2._2)

  } yield (e1._1, e1._2, e2._2)
```

Ex 1.2

```
def triangles(edges: DirectedGraph): List[(NodeId, NodeId, NodeId)] =
  edges.filter { e1 =>
    e1._1 < e1._2
  }.flatMap { e1 =>
    edges.filter { e2 =>
      e1._2 == e2._1 && edges.contains((e2._2, e1._1))
    }.filter { e2 =>
      e1._1 < e2._2 && e2._1 != e2._2
    }.map { e2 =>
      (e1._1, e1._2, e2._2)
    }
  }
```