This weeks assignments are going to cement our knowledge of language and FP, get acquainted with some of the facilities for parallel and asynchronous programming in Scala. For this weeks assignments you're required to provide unit-test coverage using ScalaTest framework. Tests should be defined in src/test/scala.

Ex1

See Traverse.scala in week3 repo (make sure to git pull).
In the below assignments you should use only for-comprehensions to iterate/filter, and groupBy/reduce to aggregate. Then write second versions of the functions - transforming for-comprehension into using map/flatMap/filter instead. You can use Tuples for return results. You can use parallel collections for parallelizing.

- Return a list of managers and number of employees for each of them
- Return list of employees with lowest salary in each department
- Return a list of employees of all departments where the sum salary (of department) is higher than a given value
- Return a list of employees which were hired before hire date of the manager of their department.

Transform above iterations so they are executed in parallel. Measure execution time using provided timed/measure utils (you can find in repository).

Ex2

Build your own parallel collection implementation for Strings. See ParCol.scala in last week's git repo and finish implementation of it. Measure how parallel version performs in comparison to regular String - use provided timed/measure utils.

Ex3

Traverse tree in parallel. Having data structure below, implement function sum which can traverse a given tree in parallel using either fork-join framework or Task parallel abstraction we built in previous class and calculate the sum of elements in the Tree.

```
sealed trait Tree /* extends Traversable */ /* with Iterable */

case class Node(value: Int, left: Tree, right: Tree) extends Tree

case object End extends Tree
```