Fun with collections

(points for each exercise lined first):

Using collections functions write utility functions that (they all need to be generic):

2 - finds repeated elements: `repeated(List(-8,5,6,1,4,4,2,5,-1,9,9))` should give: 5, 4, 9

1 - finds all elements that are not repeated

1 - finds all elements that are consecutively repeated, here: 4, 9

3 - higher order that merges two collections in a selective way, that given two collections and a partial function processes pair of objects and if they satisfy predicate, applies transformation to compute object to be put in the output collection: e.g:

```
condMerge(List(2,-9,1,8), Vector(3,-2,45,2), { case x: Tuple2[Int,Int] if max(x._1, x._2) > 0 =>
max(x._1,x._2) } ) should give: List(3, 45, 8)
```

3 - higher order that selects element from the collection given desired values and function to obtain these values. e.g.:

```
case class Person(val id: Int, val age: Int, name: String);

val team = List(Person(0, 25, "Jerry"), Person(1, 34, "Jane"), Person(2, 25,"Jim"), Person(3,
19,"Judith"))

  println(select_from(team, (x: Person) => x.id, 1, 3)) // gives: List(Person(1,34,Jane),
Person(3,19,Judith))
  println(select_from(team, (x:Person) => x.age, 25 )) // gives: List(Person(0,25,Jerry),
Person(2,25,Jim))
```