Project 2: Cities and Rivers

The Mondial database contains information about cities and rivers. The *located* table indicates on which water body (river, lake or sea) cities are located. The *river* table indicates into which water body each river flows.

The goal of the project is to write an application that, given a city C located on a river R finds all the cities from which one can reach C by navigating on R or by navigating on rivers that flow (directly or indirectly) into R.

For example, if C is `Geneva`, one can reach `Geneva` from `Sion` by navigating on the `Rhone`, or from `Lyon` that is on the `Saone` that flows into the `Rhone`, or from `Besancon` that is on the `Doubs` that flows into the `Saone` that flows into the `Rhone`.

Specification

Your program must

1. read the name of a city (from the terminal)
2. if the city is on a river print
   - the cities that are on the same river
   - the cities that are on a river that flows into this river
   - the cities that are on a river that flows into a river that flows into this river
   - etc.

Technique

To query the Mondial database from a Scala program you can adapt the following piece of code:

```
import java.net._
// a method to find all the cities on river r
def citiesOnRiver(r: String): Set[String] = {
  val q = "select city from located where river = '" + r + "'"
  val eq = URLEncoder.encode(q, "UTF-8")
  val u = new java.net.URL(
    "http://kr.unige.ch/phpmyadmin/query.php?db=Mondial"+"&sql="+eq)
  val in = scala.io.Source.fromURL(u, "iso-8859-1")
  var res = Set[String]()
  for (line <- in.getLines) {
    val cols = line.split("\t")
    res += cols(0)
  }
  in.close()
  res
}
// test
val r = citiesOnRiver("Rhone")
println(r)
```

Extra bonus

You can improve your application by taking the lakes into account. For instance, `Lausanne` is not on the `Rhone` river but on `Lac Leman` that flows into `Rhone`. Thus `Geneva` can be reached from `Lausanne`. In addition some rivers do not flow into a river but into a lake that flows into another river. In the *river* table if *River* and *Sea* are null and *Lake* is not null then *Lake* is the final destination of the river.