CS3010-2 Scala, Fall 2014: Problem Set 1

Daniel Bauer

Total points: 20
Due date: Oct 28, 11:59pm EST

Submission instructions:
Place the files for all problems in a directory named `[your_uni]_week[`X`]`, where X is the number of the problem set. For instance if your uni is xy1234 and you are submitting the problem set for the first week, the directory should be called `xy1234_week1`. Either zip or tar and gzip the directory (using `tar -c xy1234_week1 | gzip > xy1234_week1.tgz`) and upload it to your directory in the drop box for this class on Courseworks.

It is often easier to experiment in the Scala REPL before writing your final solution into a file.

All `.scala` files in this problem set contains *scripts* that can be run directly using `scala filename.scala`.

Please pay attention to the general guidelines/homework policy on the course website.

Part 0 (0 points) - Setting up Scala

1. Make sure your computer has a Java runtime environment (version 1.6 or later). Open a terminal (on Windows: open a command prompt) and type `java -version`. You should see something like this:

   ```
   $ java -version
   java version "1.7.0_51"
   Java(TM) SE Runtime Environment (build 1.7.0_51-b13)
   Java HotSpot(TM) 64-Bit Server VM (build 24.51-b03, mixed mode)
   ```

   If you do not have Java installed, you can download the Java SDK here: `http://www.oracle.com/technetwork/java/javase/downloads`.

2. Install Scala:

   - Download Scala 2.11 from `http://scala-lang.org/download/`.
   - Unpack the archive into any directory (for instance, `/usr/local/share/scala` on your Mac or `C:\Progra~1\scala` on Windows).
   - Set the `SCALA_HOME` environment variable to this directory. Then add the `SCALA_HOME/bin` subdirectory to your PATH environment variable [1].
   - Test the installaton by opening a command line (or Windows command prompt) and running `scala -version`. You should see something like this:

     ```
     $ scala -version
     Scala code runner version 2.11.2 -- Copyright 2002-2013, LAMP/EPFL
     ```

[1] See `http://environmentvariables.org/Getting_and_setting_environment_variables`

Part 1 (8 points) - Longest Collatz Sequence

Using the function `collatz` in the file `Collatz.scala` (available on the course website), write another function `longest_collatz`, that takes an integer n as its parameter and return the positive integer m, $m \leq n$ for which `collatz` needs the largest number of steps. Add your function to the file `Collatz.scala`.

Answer the following questions as comments at the end of `Collatz.scala`:

- Which number m, $m \leq 1000$, produces the longest Collatz sequence? How many steps are in the sequence?
- For very large n (e.g. $n = 1,000,000$) the naive implementation of `longest_collatz` becomes very slow. Why? Describe in words how you could improve the function to terminate faster (you do not have to implement a better solution. The naive one is fine).

Part 2 (8 points) - Lists

(a) Write a function rotate that takes a `List` of integers and returns a new `List` that has been "rotated" left.

For instance:

```
scala> val x = rotate(1 :: 2 :: 3 :: 4 :: 5 :: Nil)
x: List[Int] = List(2, 3, 4, 5, 1)

scala> rotate(x)
res1: List[Int] = List(3, 4, 5, 1, 2)
```

Hint:

- `Lists` have a method `head`, that returns the first element of the list. E.g:

  ```
  scala> List(42,23,5).head
  res0: Int = 42
  ```

- `Lists` also have a method tail that returns a `List` (of the same type) containing all elements except for the first one. E.g:

  ```
  scala> List(42,23,5).tail
  res1: List[Int] = List(23, 5)
  ```

(b) write a recursive function `rotate_n`, that takes a `List` of integers and an integer n as parameters, and rotates the list n times.

Add your code to the file `Rotate.scala`, which is available on the course website.

Part 3 (4 points) - Types

Briefly describe a situation in which the following code would make sense (and would not return an exception).

```
scala> x = y = 1
```

Hint: Think of the variable types.

Submit your explanation in a separate file called `part3.txt`.