CSCI 3155: Lab Assignment 1

Spring 2013: Due Saturday, February 2, 2012 6:00 pm

The purpose of this assignment is to warm-up with Scala and to refresh preliminaries from prior courses.

Find a partner. You will work on this assignment in pairs. Feel free to use the Piazza forum to locate a partner. However, note that **each student needs to submit a write-up** and are individually responsible for completing the assignment.

You are welcome to talk about these questions in larger groups. However, we ask that you write up your answers in pairs. Also, be sure to acknowledge those with which you discussed, including your partner and those outside of your pair.

Recall the evaluation guideline from the course syllabus.

> *Both your ideas and also the clarity with which they are expressed matter—both in your English prose and your code!*
>
> *We will consider the following criteria in our grading:*
>
> - How well does your submission answer the questions? *For example, a common mistake is to give an example when a question asks for an explanation. An example may be useful in your explanation, but it should not take the place of the explanation.*
> - How clear is your submission? *If we cannot understand what you are trying to say, then we cannot give you points for it. Try reading your answer aloud to yourself or a friend; this technique is often a great way to identify holes in your reasoning. For code, not every program that "works" deserves full credit. We must be able to read and understand your intent. Make sure you state any preconditions or invariants for your functions (either in comments, as assertions, or as `require` clauses as appropriate).*

Try to make your code as concise and clear as possible. Challenge yourself to find the most crisp, concise way of expressing the intended computation. This may mean using ways of expression computation currently unfamilar to you. Line counts for each function are given for an instructor solution (that uses good style and does not try optimize for length). While line counts are not an ideal measure of crisp, concise code, the figures give you a sense of what can be done. Can you beat the numbers while maintaining good style?

Finally, make sure that your file compiles and runs (using Scala 2.9.2 or Scala 2.10.0). A program that does not compile will *not* be graded.

Submission Instructions. First, submit your `.scala` file to the auto-testing system to get instant feedback on your code. You may submit as many times as you want, but please submit at least once before the deadline. We will only look at your last submission. A link to the auto-testing system is available on Piazza.

Then, submit to github the following:

- `Lab1-`*YourIdentiKey*`.md` with your answers to the written questions.
- `Lab1-`*YourIdentiKey*`.scala` with your answers to the coding exercises. Each member of the team needs to have a pull request.

Replace *YourIdentiKey* with your IdentiKey (e.g., for me, I would submit `Lab1-bec.md` and `Lab1-bec.scala`). Don't use your student identification number. To help with managing the submissions, we ask that you rename your uploaded files in this manner. Note that we ask you to submit your final `.scala` file to github in addition to the auto-testing system.

Getting Started. Download the code template `Lab1.scala` from the assignment page. Be sure to also write your name, your partner, and collaborators there.

1. **Feedback**. Complete the survey for this lab by filling out the answers to SURVEY.md in the Lab1/ folder in the repository. Any non-empty answer will receive full credit.

2. **Scala Basics: Binding and Scope**. For each the following uses of names, give the line where that name is bound. Briefly explain your reasoning (in no more than 1-2 sentences).

   (a) Consider the following Scala code.

   ```scala
   1  val pi = 3.14
   2  def circumference(r: Double): Double = {
   3    val pi = 3.14159
   4    2.0 * pi * r
   5  }
   6  def area(r: Double): Double =
   7    pi * r * r
   ```

   The use of `pi` at line 4 is bound at which line? The use of `pi` at line 7 is bound at which line?

   (b) Consider the following Scala code.

   ```scala
   1  val x = 3
   2  def f(x: Int): Int =
   3    x match {
   4      case 0 => 0
   5      case x => {
   6        val y = x + 1
   7        ({
   8           val x = y + 1
   ```

```
9            y
10         } * f(x - 1))
11       }
12     }
13   val y = x + f(x)
```

The use of `x` at line 3 is bound at which line? The use of `x` at line 6 is bound at which line? The use of `x` at line 10 is bound at which line? The use of `x` at line 13 is bound at which line?

3. **Scala Basics: Typing**. In the following, I have left off the return type of function `g`. The body of `g` is well-typed if we can come up with a valid return type. Is the body of `g` well-typed?

```
def g(x: Int) = {
  val (a, b) = (1, (x, 3))
  if (x == 0) (b, 1) else (b, a + 2)
}
```

If so, give the return type of `g` and explain how you determined this type. For this explaination, first, give the types for the names `a` and `b`. Then, explain the body expression using the following format:

$e : \tau$ because

  $e_1 : \tau_1$ because

    ...

  $e_2 : \tau_2$ because

    ...

where e_1 and e_2 are subexpressions of e. Stop when you reach values (or names).

As an example of the suggested format, consider the `plus` function:

```
def plus(x: Int, y: Int) = x + y
```

Yes, the body expression of `plus` is well-typed with type `Int`.

`x + y` : `Int` because

  `x` : `Int`

  `y` : `Int`

4. **Some Library Functions**. Most languages come with a standard library with support for things like data structures, mathematical operators, string processing, etc. For this question, we will implement some library operations.

For each function f that you are asked to write in this question, write one test case as a **def** named `test`f that returns a `Boolean` (with the first letter capitalized) that is an check between an expected result and the computed result. One can then use an `assert` to make sure your implementation passes your unit test. For example,

```
def plus(x: Int, y: Int): Int = x + y
def testPlus(plus: (Int, Int) => Int): Boolean = plus(1,1) == 2
assert(testPlus(plus))
```

(a) Write a function `abs`

```
def abs(n: Double): Double
```

that returns the absolute value of `n`. This a function that takes a value of type `Double` and returns a value of type `Double`. This function corresponds to the JavaScript library function `Math.abs`.

Instructor Solution: 1 line.

(b) Write a function `xor`

```
def xor(a: Boolean, b: Boolean): Boolean
```

that returns the exclusive-or of `a` and `b`. The exclusive-or returns **true** if and only if exactly one of `a` or `b` is **true**. For practice, do not use the Boolean operators. Instead, only use the **if-else** expression and the Boolean literals (i.e., **true** or **false**).

Instructor Solution: 4 lines (including 1 line for a closing brace).

5. **Recursion**.

(a) Write a recursive function `repeat`

```
def repeat(s: String, n: Int): String
```

where `repeat(s, n)` returns a string with `n` copies of `s` concatenated together. For example, `repeat("a",3)` returns `"aaa"`. This function corresponds to the Google Closure library function `goog.string.repeat`.

Instructor Solution: 4 lines (including 1 line for a closing brace).

(b) In this exercise, we will implement the square root function—`Math.sqrt` in the JavaScript standard library—using Newton's method (also known as Newton-Raphson).

Recall from Calculus that a root of a differentiable function can be iteratively approximated by following tangent lines. More precisely, let f be a differentialable function, and let x_0 be an initial guess for a root of f. Then, Newton's method specifies a sequence of approximations $x_0, x_1, \dots$ with the following recursive equation:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \ .$$

The following link is a refresher video on this algorithm: `http://www.youtube.com/watch?v=1uN8cBGVpfs`.

The square root of a real number c for $c > 0$, written $\sqrt{c}$, is a positive x such that $x^2 = c$. Thus, to compute the square root of a number c, we want to find the positive root of the function:

$$f(x) = x^2 - c \ .$$

Thus, the following recursive equation defines a sequence of approximations for $\sqrt{c}$:

$$x_{n+1} = x_n - \frac{x_n^2 - c}{2x_n} .$$

i. First, implement a function `sqrtStep`

```
def sqrtStep(c: Double, xn: Double): Double
```

that takes one step of approximation in computing $\sqrt{c}$ (i.e., computes x_{n+1} from x_n).
Instructor Solution: 1 line.

ii. Next, implement a function `sqrtN`

```
def sqrtN(c: Double, x0: Double, n: Int): Double
```

that computes the nth approximation x_n from an initial guess x_0. You will want to call `sqrtStep` implemented in the previous part.
Challenge yourself to implement this function using recursion and no mutable variables (i.e., **var**s)—you will want to use a recursive helper function. It is also quite informative to compare your recursive solution with one using a **while** loop.
Instructor Solution: 8 lines (including 2 lines for closing braces and 1 line for a `require`).

iii. Now, implement a function `sqrtErr`

```
def sqrtErr(c: Double, x0: Double, epsilon: Double): Double
```

that is very similar to `sqrtN` but instead computes approximations x_n until the approximation error is within ε (`epsilon`), that is,

$$|x_n^2 - c| < \varepsilon .$$

You can use your absolute value function `abs` implemented in a previous part.
A wrapper function `sqrt` is given in the template that simply calls `sqrtErr` with a choice of `x0` and `epsilon`.
Again, challenge yourself to implement this function using recursion and compare your recursive solution to one with a **while** loop.
Instructor Solution: 8 lines (including 2 lines for closing braces and 1 line for a `require`).

6. **Data Structures**. In this question, we will review implementing operations on binary search trees from Data Structures. Balanced binary search trees are common in standard libraries to implement collections, such as sets or maps. For example, the Google Closure library has `goog.structs.AvlTree`. For simplicity, we will not worry about balancing in this question.

   Trees are important structures in developing interpreters, so this question is also critical practice in implementing tree manipulations.

A binary search tree is a binary tree that satisfies an ordering invariant. Let n be any node in a binary search tree whose data value is d, left child is l, and right child is r. The ordering invariant is that all of the data values in the subtree rooted at l must be $< d$, and all of the data values in the subtree rooted at r must be $\geq d$.

We will represent a binary search tree using the following Scala classes and objects:

```
sealed abstract class SearchTree
case object Empty extends SearchTree
case class Node(l: SearchTree, d: Int, r: SearchTree) extends SearchTree
```

A `SearchTree` is either `Empty` or a `Node` with left child `l`, data value `d`, and right child `r`.

For this question, we will implement the following four functions.

(a) The function `repOk`

```
def repOk(t: SearchTree): Boolean
```

checks that an instance of `SearchTree` is valid binary search tree. In other words, it checks using a traversal of the tree the ordering invariant. This function is useful for testing your implementation. A skeleton of this function has been provided for you in the template.

Instructor Solution: 7 lines (including 2 lines for closing braces).

(b) The function `insert`

```
def insert(t: SearchTree, n: Int): SearchTree
```

inserts an integer into the binary search tree. Observe that the return type of `insert` is a `SearchTree`. This choice suggests a functional style where we construct and return a new output tree that is the input tree `t` with the additional integer `n` as opposed to destructively updating the input tree.

Instructor Solution: 4 lines (including 1 line for a closing brace).

(c) The function `deleteMin`

```
def deleteMin(t: SearchTree): (SearchTree, Int)
```

deletes the smallest data element in the search tree (i.e., the leftmost node). It returns both the updated tree and the data value of the deleted node. This function is intended as a helper function for the `delete` function. Most of this function is provided in the template.

Instructor Solution: 9 lines (including 2 lines for closing braces).

(d) The function `delete`

```
def delete(t: SearchTree, n: Int): SearchTree
```

removes the first node with data value equal to `n`. This function is trickier than `insert` because what should be done depends on whether the node to be deleted has children or not. We advise that you take advantage of pattern matching to organize the cases.

Instructor Solution: 10 lines (including 1 line for a closing brace).