Hello, Scala

Due: Monday, September 17, 2012 at 11:59pm Pacific
Complete this assignment on your own — 200 total points

Overview

In class, we briefly explored Scala as a group. This assignment asks you to explore Scala some more on your own. The main goals of this assignment are: (a) to help you become more familiar with Scala's support for data structures and operations that are common to most general-purpose languages and (b) to become acquainted with the various sources of help and documentation for Scala programming. The best places to start are in the required book for this course, in the other Scala resources listed in the syllabus, and in the Scala documentation available online.

1. Numeric and collection types *(100 total points)*

This part of the assignment asks you to explore more of Scala's basic data types. Do your work in an interactive Scala session, discover expressions and results that best answer the questions below, and paste the expressions and results into the file `Answers.txt`.

The "Materials" section contains information on where to get the files such as `Answers.txt`.

Basic numeric types *(6 * 8 points = 48 points)*
For each of the basic numeric types listed below, provide: (i) the minimum value allowed for that type, (ii) the maximum value allowed for that type, (iii) an expression that illustrates an operation on that type, (iv) the expression's result, and (v) an English explanation of the operation.

You can look up the type in the Scala API. Be sure to use the API that corresponds to your version of Scala.

An example solution for `Int` might be:

> minimum value: `-2^31`
> maximum value: `2^31 - 1`
> operation and result:
> ```
> scala> 1 max 3
> res4: Int = 3
> ```
> description: The `max` operator takes two integer arguments and returns the argument that has the larger value.

Try to find an operation whose meaning wasn't obvious to you at first.

Some types have the same operations—be sure to describe a different operation for each answer. Fill in `Answers.txt` for the following numeric types:

Each basic numeric type also has a "rich wrapper" type (e.g., `RichInt`) that provides more operations. You can choose an operations from these, too.

(a) `Byte`
(b) `Short`
(c) `Long`
(d) `Char`
(e) `Float`
(f) `Double`

Big numbers *(4 points)*
Here's what happens when you try to create a number that's too big for the basic types:

```
scala> val bigint = 10000000000000000000000000000000
<console>:1: error: integer number too large
```

Fill in `Answers.txt` with a Scala statement that answers the following question:

(g) How can we create a `val` that holds a large number?

Find an expression that will represent a value that's too big even for a `Double`.

(Part 2 continued on next page...)

Collection types *(4 * 8 points = 32 points)*
Each of the statements below defines a `val` whose value is a collection. For each `val`, provide (i) an expression that retrieves an element from the collection, (ii) the expression's result, (iii) an expression that illustrates an operation on the `val`, (iv) the expression's result, and (v) an English explanation of the operation.

For example, given the following definition of an array value:

```
val myArray = Array("a", "b", "c")
```

an example solution might be:

element retrieval and result:
```
scala> myArray(0)
res6: java.lang.String = a
```
element operation and result:
```
scala> myArray.length
res8: Int = 3
```
description: length gives the length of the array (duh).

Fill in `Answers.txt` for the following `val`s:

(h) `val myList = List(1, 2, 3)`

(i) `val myTuple = ("a", 1)`

(j) `val mySet = Set(1, 1, 2)`

(k) `val myMap = Map("Scala" -> 2003, "Python" -> 1991)`

Equality *(8 points)*
Based on the following interactive session:

```
scala> var set1 = Set(1,2,3)
set1: scala.collection.immutable.Set[Int]=Set(1, 2, 3)

scala> var set2 = Set(1,1,2,3)
set2: scala.collection.immutable.Set[Int]=Set(1, 2, 3)

scala> set1 == set2
res18: Boolean = true

scala> set1 eq set2
res19: Boolean = false
```

answer the following question:

(l) What's the difference between `==` and `eq`?

(Part 2 continued on next page...)

Immutability *(8 points)*
Sets are immutable by default, which means that you can't change a set's value. The assignment to `set1` in the following interactive session seems to contradict the notion that sets are immutable:

```
scala> var set1 = Set(1,2,3)
set1: scala.collection.immutable.Set[Int] = Set(1,2,3)

scala> var set1r = set1
set1r: scala.collection.immutable.Set[Int] = Set(1,2,3)

scala> set1r == set1
res0: Boolean = true

scala> set1r eq set1
res1: Boolean = true

scala> set1 += 4

scala> set1
res3: scala.collection.immutable.Set[Int]=Set(1,2,3,4)

scala> set1r
res5: scala.collection.immutable.Set[Int] = Set(1,2,3)

scala> set1r eq set1
res6: Boolean = false
```

(m) What happens when Scala executes `set1 += 4` (i.e., why doesn't this assignment contradict the fact that sets are immutable)?

2. Strings *(5 * 6 points = 30 total points)*

The `StringOps` class could be helpful here.

Use Scala's string operations to answer the following questions. Answer questions (a)–(e) by filling in the body of the corresponding function in the file `strings.scala`. Answer question (f) by filling in the comment at the bottom of the file `strings.scala`.

(a) Produce the following value:

```
Array("Gallop apace, you fiery-footed steeds,",
"Towards Phoebus' lodging: such a wagoner",
"As Phaethon would whip you to the west,",
"And bring in cloudy night immediately.")
```

(b) Print each element of `designers`, capitalized, on its own line.

(c): *Note that whitespace has been stripped from the array's elements.*

(c) Produce the following value:

```
Array("cherries", "mangos in syrup", "bananas")
```

(d) Use only one `String` operation and one `List` method call to produce the following value:

```
"TAs = Odersky + Wirth + Hoare + Liskov + Backus + Gosling"
```

(e) For each key, value pair in `rates`, print each pair on its own line, separated by the string `": "`, with the key in a string of length 9, and the value accurate to two decimal places.

(f) Why do the functions `question_b` and `question_e` have result type `Unit`?

3. Functions *(3 * 12 points = 36 total points)*

Place your answers in the file `functions.scala`, where indicated.

Function definition syntax

The following function definition is a slight modification of one from class:

```
def sumListIterative2(list: List[Int]) {
    var sum = 0
    for (item <- list)
        sum += item
    sum
}
```

When called with the following expression:

```
sumListIterative2(List(1,2,3,4))
```

The result is `()` instead of the expected value of `10`.

(a) Why was the result `()`? What change(s) to the function definition should you make to get the expected behavior?

Fibonacci

Let the Fibonacci numbers be defined by the following equation:

Note that the function as given is undefined for negative numbers. Still, it should result in a value rather than diverge.

$$fib(n) = \begin{cases} 0 & : n = 0 \\ 1 & : n = 1 \\ fib(n-2) + fib(n-1) & : n \geq 2 \end{cases}$$

(b) Write a function `fib_recursive` that takes an integer, results in an integer, and computes the Fibonacci numbers recursively.

(c) Write a function `fib_iter` that takes an integer, results in an integer, and computes the Fibonacci numbers iteratively.

4. Codehunt *(34 total points)*

The file `enc.txt` contains some text "encrypted" in binary. The file `decode.scala` contains the following definitions:

- `data`: a val that contains input from `stdin`. The input is assumed to be encoded as a binary string (i.e., a string that contains only the characters `'1'` and `'0'`).
- `partitionByteStrings`: a function that takes a binary string and partitions it into a list of *byte strings*. A byte string is a string of length 8, whose information corresponds to one byte. For example: `"01100001"`.
- `byteString2Char`: a function that takes a byte string and converts it to its corresponding `Char`. For example `byteString2Char("01100001")` should produce the value `'a'`.
- `byteStrings2Chars`: a function that takes a list of byte strings and produces a list of `Char`s. (This function lifts `byteString2Char` to operate over lists.)
- `charsToString`: A function that takes a list of `Char`s and joins them into a single `String`.
- `decode`: a function that takes a binary string and produces the plaintext version.

`decode.scala` imports `scala.math.pow`, which may be helpful in implementing this function.

(a) Fill in the empty function definitions in `decode.scala`. It should be possible to type the following shell command and see the corresponding decoded text:

```
cat enc.txt | scala decode.scala
```

(17 points)

(b) The encrypted text is from a blog post. What is the title of the blog post? Place your answer where indicated in `decode.scala`. *(5 points)*

(c) The file `encode.scala` should invert the operations in `decode.scala`. Fill in the empty function definitions to implement the inversion. It should be possible to type the following shell command and receive no output:

```
cat enc.txt | scala decode.scala | scala encode.scala | diff enc.txt -
```

(12 points)

Feedback *(participation points)*

Answer the questions in the file `feedback.txt`.

Materials

The materials are available in a Subversion repository. To retrieve them, perform the following commands, substituting your knuth user id for `<id>` and the name of the local directory where you'll do your work for `<dir>`:

```
cd <dir>

svn mkdir https://svn.cs.hmc.edu/dsl/fall12/ours/<id> -m 'creating my directory'

svn co https://svn.cs.hmc.edu/dsl/fall12/ours/<id>

cd <id>

svn copy https://svn.cs.hmc.edu/dsl/fall12/hw/3 hw3
```

The materials should now be in `<dir>/<id>/hw3/`. Modify the materials to supply your answers, and use the instructions below to submit. Don't forget to add your information to the top of every file, where indicated.

Submitting Your Solution

To submit, make sure you're in the `<dir>/hw3` directory, and type the following command:

```
svn commit
```

Committing is like voting in Chicago: you should do it early and often.

Collaboration and Honor Code

I expect you to abide by the Harvey Mudd Honor code. Your solution to this assignment should be produced by you alone. You may discuss concepts at a high level with any student in the class. However, you may not copy solutions from anyone.

If you have any questions about what behavior is acceptable, it is your responsibility to come see me before you engage in this behavior. I will be happy to answer any of your questions.