Assignment #1

CSE341: Principles of Programming Languages
Hyungon Moon

Out: Sep 15, 2020 (Tue)
Due: Sep 28, 2020 (Mon), 23:59 (KST)

Rules

- You must not use the `var`, `for`, or `while` keyword.
- You must not include any additional packages or librariles besides the ones that you already have.

Scala environment

1. Check if you have java using this command in a terminal (Windows: PowerShell, Mac/Linux: Terminal). You can use the one through VS Code also.

   Command Line
   ```
   java -version
   ```

   If you have java 11, you'll see messages like these.

   Command Line
   ```
   openjdk version "11.0.8" 2020-07-14
   OpenJDK Runtime Environment AdoptOpenJDK (build 11.0.8+10)
   OpenJDK 64-Bit Server VM AdoptOpenJDK (build 11.0.8+10, mixed mode)
   ```

   Command Line
   ```
   openjdk version "11.0.2" 2019-01-15
   OpenJDK Runtime Environment 18.9 (build 11.0.2+9)
   OpenJDK 64-Bit Server VM 18.9 (build 11.0.2+9, mixed mode)
   ```

2. If you don't have java 11, install it. You can download and install here.
   https://adoptopenjdk.net/
   Download and install OpenJDK 11(LTS), HotSpot

3. Now you can work with the assignment. Using a terminal, run sbt using these commands.
   On Windows,

Command Line

```
sbt/bin/sbt.bat
```

On Mac/Linux,

Command Line

```
sbt/bin/sbt
```

It could take long time to collect all necessary files from the internet when you first run the command.

4. Once everything has been collected, you will get an sbt shell access. You can type commands here to test your submission. Typing `test` will test the submission using accompanied test cases.

Command Line

```
sbt:hw1-cse341> test
```

Problems

Problem 1 (10 points)

Fibonacci sequence is a sequence of numbers each of which is the sum of the two preceding one. We can define the n^{th} Fibonacci numbers F_n as follows.

$$F_n = \begin{cases} 1 & \text{if } n < 2 \\ F_{n-1} + F_{n-2} & \text{otherwise} \end{cases} \tag{1}$$

Write a function fibo that evaluates to the nth fibonacci number:

```
def fibo (n: Int): Int
```

Problem 2 (10 points)

Write a function

```
def sum(f: Int=>Int, n: Int): Int
```

that takes another function f and in integer n, and computes the sum of values obtained by applying $1, 2, 3, ...n$ to f.
In other words,

```
sum(f,n)
```

computes

$$\sum_{k=1}^{n} f(k) \tag{2}$$

For example,

```
sum((x: Int) => x, 10)
```

evaluates to 55.

Problem 3 (10 points)

Take this from the lecture slide as the definition of a list data structure.

```
sealed trait IntList
case object Nil extends MyList
case class Cons(v: Int, t: MyList) extends MyList
```

Write a function

```
def foldRight(init: Int, ftn: Int=>Int, list: IntList): Int
```

that takes a function from integers to integers and a list as an input, and folds to right. For example,

```
foldRight(100, (x: Int, y:Int) => x % y, Cons(5,Cons(3,Nil))
```

evaluates to $(100 \,\%\, 3) \,\%\, 5 \;=\; 1$ while

```
foldRight(100, (x: Int, y:Int) => x % y, Cons(3,Cons(5,Nil))
```

evaluates to $(100 \,\%\, 5) \,\%\, 3 \;=\; 0$

Problem 4 (10 points)

Write a function with type:

```
def filter(f: Int => Boolean, list: IntList): IntList
```

that construct a list from another list by collecting only the elements that passes the test `f`, i.e., `f(x)` evaluates tot `true`.
For example,

```
filter((x: Int) => x % 3 == 0, Cons(5,Cons(3,Cons(6,Nil))))
```

evaluates to `Cons(3,Cons(6,Nil))`.

Problem 5 (10 points)

Write a function with type:

```
def iter[A](f: A => A, n: Int): A => A
```

that computes a composition of another function by the given number of times, like this.

$$\begin{aligned} \text{iter}(f,1) &= f \\ \text{iter}(f,2) &= f \circ f = f^2 \\ &\cdots \\ \text{iter}(f,n) &= f^n \end{aligned} \tag{3}$$

Problem 6 (20 points)

Write a function that inserts a new integer to a binary search tree. Binary search tree is a binary tree in which all left children are smaller than the parent and all right children are larger than the parent.Use the following as the definition of the tree:

```
sealed trait BTree
case object Leaf extends BTree
case class IntNode(v: Int, left: BTree, right: BTree)
extends BTree
```

And write a function of type:

```
def insert(t: BTree, a: Int): BTree
```

Assume that the incoming tree does not have any duplicates, and don't insert if the number to be inserted is already in the tree.

Problem 7 (30 points)

Write a function that evaluates a formula into a truth value. The set of formula is defined as follows:

$$\begin{aligned} f \rightarrow & \; T | F \\ & | \; !f \\ & | \; f \; \&\& \; f \\ & | \; f \; || \; f \\ & | \; f \; \text{->} \; f \end{aligned} \tag{4}$$

The following code snippet defines the types representing the formular.

```
sealed trait Formula
case object True extends Formula
case object False extends Formula
case class Not(f: Formula) extends Formula
case class Andalso(left: Formula, right: Formula) extends Formula
case class Orelse(left: Formula, right: Formula)  extends Formula
case class Implies(left: Formula, right: Formula) extends Formula
```

Thus, the function will have the type:

```
def eval(f: Formula): Boolean
```

For example,

```
eval(True())
```

evaluates to `true`.

```
eval(Andalso(Not(True()),OrElse(Not(True(),True)))
```

evaluates to `false`.