Bowen Jiang (NUID: 001582174)

CSYE 7200

Big-Data Sys Engr Using Scala

Sprint 2021

Assignment 4(Random State)

Task:

We need to create a trait called *RandomState* which will have two obvious methods: *next* and *get*. Of course, we don't really know what the type of the result of *get* will be, so let's make it parametric, thus: *RandomState[T]*.
But once we have a *RandomState[T]*, we will want to be able to map it into a *RandomState[U]* so we'll need to implement *map*. While we're at it, we might as well implement *flatMap* too. Technically, this will mean that it's a "monad" but we haven't talked about those yet -- but they are important.
There's one other convenience method that we should probably implement and that is *toStream* which will return a *LazyList[T]*. As usual, I have provided the basic framework and a specification for your work: *src/main/scala/edu/neu/coe/csye7200/asstrs/RandomState.scala* and the corresponding *RandomStateSpec* in the *test* directory. All you have to do is to implement the 6 *TO BE IMPLEMENTED* and run the tests. When it's all green, you're done. You can get these from the class repo (see *Course Material/Resources/Class Repository*), the module name for this assignment is *assignment-random-state*.

Function Implement

The function I implement:

```
// Hint: Think of the input and output, find the appropriate method that achieve this.
// 10 points
def flatMap[U](f: T => RandomState[U]): RandomState[U] = f(get) // TO BE IMPLEMENTED

// Hint: This a recursively method and it concatenate current element with following elements.
// 12 points
def toStream: LazyList[T] = LazyList.cons[T](get, next.toStream) // TO BE IMPLEMENTED

var ran = new Random(n)
// Hint: Remember to use the "seed" to generate next RandomState.
// 7 points
def next: RandomState[T] = JavaRandomState[T](ran.nextLong(), g) // TO BE IMPLEMENTED
// Hint: Think of the input and output.
// 5 points
def get: T = g(n) // TO BE IMPLEMENTED
// Hint: This one need function composition.
// 13 points
def map[U](f: T => U): RandomState[U] = JavaRandomState[U](n, g.andThen(f)) // TO BE IMPLEMENTED

// Hint: This is a easy one, remember that it not only convert a Long to a Double but also scale down the number to -1 ~ 1.
// 4 points
val longToDouble: Long => Double = {x => x.toDouble/Long.MaxValue} // TO BE IMPLEMENTED
```

Unit Test Screenshot