

Anirudha Joshi – SEC01 (NUID 002991365)

Big Data System Engineering with Scala  
Spring 2023  
Assignment No. 5 (Functional  
Composition)



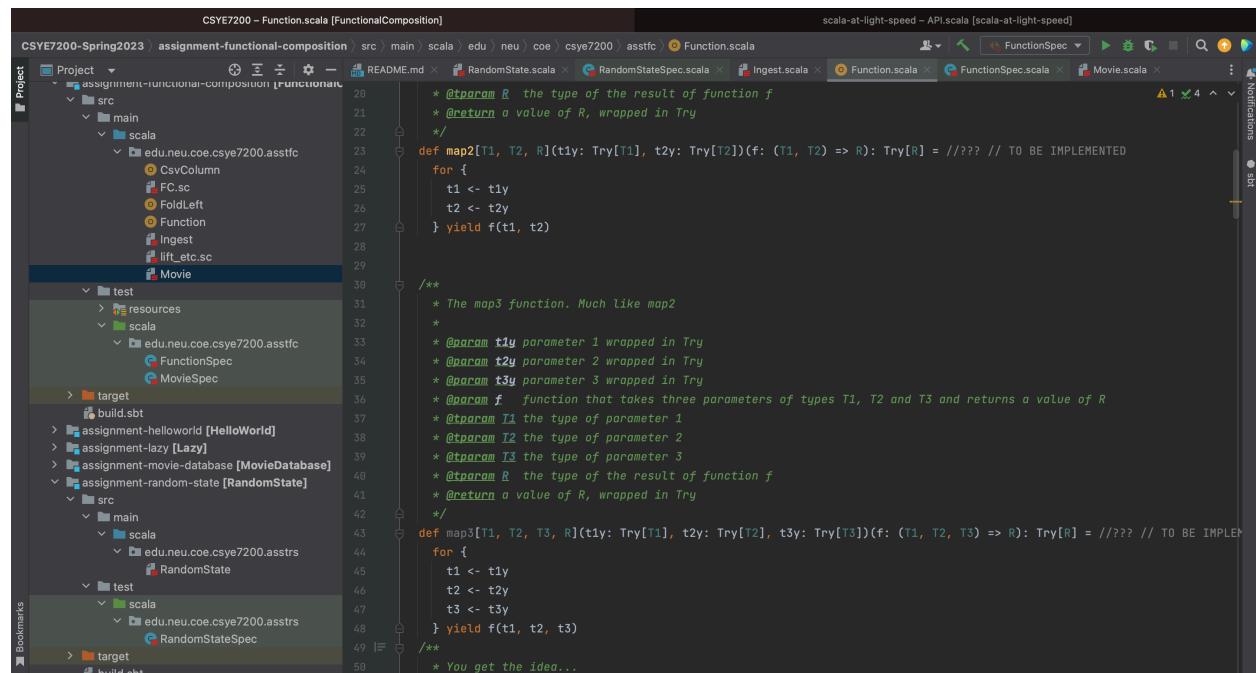
GitHub - [https://github.com/anirudhajoshi2808/Anirudha\\_Joshi\\_CSYE7200](https://github.com/anirudhajoshi2808/Anirudha_Joshi_CSYE7200)

### - Questions:

There are 13 TODOs in *Function.scala* and 2 TODOs in *Movie.scala*. You can get these from the class repo (see *Course Material/Resources/Class Repository*), the module name for this assignment is *assignment-functional-composition*.

### - Code

#### Function.scala



The screenshot shows a Scala IDE interface with the following details:

- Project Navigation:** The left sidebar shows a project structure for "CSYE7200-Spring2023". It includes several modules:
  - assignment-functional-composition**: Contains `src/main/scala/edu/neu/coe/csy7200/asstfc` with files like `CsvColumn`, `FC.sc`, `FoldLeft`, `Function`, `Ingest`, `lift_etc.sc`, and `Movie`.
  - assignment-helloworld**
  - assignment-lazy**
  - assignment-movie-database**
  - assignment-random-state**: Contains `src/main/scala/edu.neu.coe.csy7200.asstsrs` with `RandomState`.
  - assignment-stateful**
  - target** and `build.sbt`.
- Code Editor:** The main window displays the `Function.scala` file. The code contains several TODO comments (marked with `/* TO BE IMPLEMENTED */`) and a note at the bottom: `/* You get the idea...`. The code is as follows:

```
20  * @tparam R the type of the result of function f
21  * @return a value of R, wrapped in Try
22  */
23  def map2[T1, T2, R](t1y: Try[T1], t2y: Try[T2])(f: (T1, T2) => R): Try[R] = //???
24  for {
25    t1 <- t1y
26    t2 <- t2y
27  } yield f(t1, t2)
28
29 /**
30  * The map3 function. Much like map2
31  *
32  * @param t1y parameter 1 wrapped in Try
33  * @param t2y parameter 2 wrapped in Try
34  * @param t3y parameter 3 wrapped in Try
35  *
36  * @param f function that takes three parameters of types T1, T2 and T3 and returns a value of R
37  * @param t1 the type of parameter 1
38  * @param t2 the type of parameter 2
39  * @param t3 the type of parameter 3
40  * @param R the type of the result of function f
41  * @return a value of R, wrapped in Try
42  */
43  def map3[T1, T2, T3, R](t1y: Try[T1], t2y: Try[T2], t3y: Try[T3])(f: (T1, T2, T3) => R): Try[R] = //???
44  for {
45    t1 <- t1y
46    t2 <- t2y
47    t3 <- t3y
48  } yield f(t1, t2, t3)
49 /**
50  * You get the idea...
```

CSYE7200 - Function.scala [FunctionalComposition]

```
CSYE7200-Spring2023 > assignment-functional-composition | src | main | scala | edu | neu | coe | csye7200 | assstfc | Function.scala
```

Project

```
assignment-functional-composition [FunctionalComposition]
  +-- src
    +-- main
      +-- scala
        +-- edu.neu.coe.csye7200.assstfc
          +-- CsvColumn
          +-- FC.sc
          +-- FoldLeft
          +-- Function
          +-- Ingest
          +-- lift_etc.sc
          +-- Movie
        +-- test
          +-- resources
          +-- scala
            +-- edu.neu.coe.csye7200.assstfc
              +-- FunctionSpec
              +-- MovieSpec
        +-- target
          +-- build.sbt
    +-- README.md
    +-- RandomState.scala
    +-- RandomStateSpec.scala
    +-- Ingest.scala
    +-- Function.scala
    +-- FunctionSpec.scala
    +-- Movie.scala
```

```
49  /**
50   * You get the idea...
51   */
52 def map7[T1, T2, T3, T4, T5, T6, T7, R](t1y: Try[T1], t2y: Try[T2], t3y: Try[T3], t4y: Try[T4], t5y: Try[T5], t6y: Try[T6],
53                                         t7y: Try[T7]): Try[R] = //??? // TO BE IMPLEMENTED
54
55   for {
56     t1 <- t1y
57     t2 <- t2y
58     t3 <- t3y
59     t4 <- t4y
60     t5 <- t5y
61     t6 <- t6y
62     t7 <- t7y
63   } yield f(t1, t2, t3, t4, t5, t6, t7)
64
65 /**
66  * Lift function to transform a function f of type T=>R into a function of type Try[T]=>Try[R]
67  *
68  * @param f the function we start with, of type T=>R
69  * @param I the type of the parameter to f
70  * @param R the type of the result of f
71  * @return a function of type Try[T]=>Try[R]
72  */
73 // You know this one
74 def lift[T, R](f: T => R): Try[T] => Try[R] = //??? // TO BE IMPLEMENTED
75
76   (t: Try[T]) => t.map(f)
77 }
```

CSYE7200 - Function.scala [FunctionalComposition]

```
CSYE7200-Spring2023 > assignment-functional-composition | src | main | scala | edu | neu | coe | csye7200 | assstfc | Function.scala
```

Project

```
assignment-functional-composition [FunctionalComposition]
  +-- src
    +-- main
      +-- scala
        +-- edu.neu.coe.csye7200.assstfc
          +-- CsvColumn
          +-- FC.sc
          +-- FoldLeft
          +-- Function
          +-- Ingest
          +-- lift_etc.sc
          +-- Movie
        +-- test
          +-- resources
          +-- scala
            +-- edu.neu.coe.csye7200.assstfc
              +-- FunctionSpec
              +-- MovieSpec
        +-- target
          +-- build.sbt
    +-- README.md
    +-- RandomState.scala
    +-- RandomStateSpec.scala
    +-- Ingest.scala
    +-- Function.scala
    +-- FunctionSpec.scala
    +-- Movie.scala
```

```
86 /**
87  * Think Simple, Elegant, Obvious
88  */
89 def lift2[T1, T2, R](f: (T1, T2) => R): (Try[T1], Try[T2]) => Try[R] = //??? // TO BE IMPLEMENTED
90
91   (t1: Try[T1], t2: Try[T2]) =>
92     for {
93       x <- t1
94       y <- t2
95     } yield f(x, y)
96
97 /**
98  * Lift function to transform a function f of type (T1,T2,T3)=>R into a function of type (Try[T1],Try[T2],Try[T3])=>Try[R]
99  *
100 * @param f the function we start with, of type (T1,T2,T3)=>R
101 * @param I1 the type of the first parameter to f
102 * @param I2 the type of the second parameter to f
103 * @param I3 the type of the third parameter to f
104 * @param R the type of the result of f
105 * @return a function of type (Try[T1],Try[T2],Try[T3])=>Try[R]
106 */
107 // If you can do lift2, you can do lift3
108 def lift3[T1, T2, T3, R](f: (T1, T2, T3) => R): (Try[T1], Try[T2], Try[T3]) => Try[R] = //??? // TO BE IMPLEMENTED
109
110   (t1: Try[T1], t2: Try[T2], t3: Try[T3]) =>
111     for {
112       x <- t1
113       y <- t2
114       z <- t3
115     } yield f(x, y, z)
```

CSYE7200 - Function.scala [FunctionalComposition]

```

CSYE7200-Spring2023 > assignment-functional-composition | src | main | scala | edu | neu | coe > csye7200 | assfc | Function.scala
Project  Project  README.md  RandomState.scala  RandomStateSpec.scala  Ingest.scala  Function.scala  FunctionSpec.scala  Movie.scala
  > assignment-functional-composition [FunctionalComposition]
    > src
      > main
        > scala
          > edu.neu.coe.csye7200.assfc
            CsvColumn
            FC.sc
            FoldLeft
            Function
            Ingest
            lift_etc.sc
            Movie
      > test
        > resources
        > scala
          > edu.neu.coe.csye7200.assfc
            FunctionSpec
            MovieSpec
    > target
      build.sbt
  > assignment-helloworld [HelloWorld]
  > assignment-lazy [Lazy]
  > assignment-movie-database [MovieDatabase]
  > assignment-random-state [RandomState]
    > src
      > main
        > scala
          > edu.neu.coe.csye7200.asstrs
            RandomState
    > test
      > scala
        > edu.neu.coe.csye7200.asstrs
          RandomStateSpec
    > target
      build.sbt

```

```

131  /**
132   * If you can do lift3, you can do lift7
133   */
134   def lift7[T1, T2, T3, T4, T5, T6, T7, R](f: (T1, T2, T3, T4, T5, T6, T7) => R):
135     Try[T1], Try[T2], Try[T3], Try[T4], Try[T5], Try[T6], Try[T7]) => Try[R] = //???
136   // TO BE IMPLEMENTED
137   (t1y, t2y, t3y, t4y, t5y, t6y, t7y) =>
138   for {
139     t1 <- t1y
140     t2 <- t2y
141     t3 <- t3y
142     t4 <- t4y
143     t5 <- t5y
144     t6 <- t6y
145     t7 <- t7y
146   } yield f(t1, t2, t3, t4, t5, t6, t7)
147 /**
148  * This method inverts the order of the first two parameters of a two-(or more-)parameter curried function.
149  *
150  * @param f the function
151  * @tparam I1 the type of the first parameter
152  * @tparam I2 the type of the second parameter
153  * @tparam R the result type
154  * @return a curried function which takes the second parameter first
155  */
156 /**
157  * Hint: think about writing an anonymous function that takes a t2, then a t1 and returns the appropriate result
158  * // Note: you won't be able to use the "_" character here because the compiler infers an ordering that you don't want
159  * def invert2[T1, T2, R](f: T1 => T2 => R): T2 => T1 => R = //???
160  * // TO BE IMPLEMENTED
161  * (t2: T2) => (t1: T1) => f(t1)(t2)
162 /**
163  */

```

CSYE7200 - Function.scala [FunctionalComposition]

```

CSYE7200-Spring2023 > assignment-functional-composition | src | main | scala | edu | neu | coe > csye7200 | assfc | Function.scala
Project  Project  README.md  RandomState.scala  RandomStateSpec.scala  Ingest.scala  Function.scala  FunctionSpec.scala  Movie.scala
  > assignment-functional-composition [FunctionalComposition]
    > src
      > main
        > scala
          > edu.neu.coe.csye7200.assfc
            CsvColumn
            FC.sc
            FoldLeft
            Function
            Ingest
            lift_etc.sc
            Movie
      > test
        > resources
        > scala
          > edu.neu.coe.csye7200.assfc
            FunctionSpec
            MovieSpec
    > target
      build.sbt
  > assignment-helloworld [HelloWorld]
  > assignment-lazy [Lazy]
  > assignment-movie-database [MovieDatabase]
  > assignment-random-state [RandomState]
    > src
      > main
        > scala
          > edu.neu.coe.csye7200.asstrs
            RandomState
    > test
      > scala
        > edu.neu.coe.csye7200.asstrs
          RandomStateSpec
    > target
      build.sbt

```

```

169 /**
170  * @param R the result type
171  * @return a curried function which takes the third parameter first, then the second, etc.
172  */
173 /**
174  * If you can do invert2, you can do this one too
175  */
176 /**
177  * This method inverts the order of the first four parameters of a four-(or more-)parameter curried function.
178  *
179  * @param f the function
180  * @tparam I1 the type of the first parameter
181  * @tparam I2 the type of the second parameter
182  * @tparam I3 the type of the third parameter
183  * @tparam I4 the type of the fourth parameter
184  * @tparam R the result type
185  * @return a curried function which takes the fourth parameter first, then the third, etc.
186  */
187 /**
188  * If you can do invert3, you can do this one too
189  */
190 /**
191  * This method uncurries the first two parameters of a three- (or more-)
192  * parameter curried function.
193  * The result is a (curried) function whose first parameter is a tuple of the first two parameters of f;
194  * whose second parameter is the third parameter, etc.
195  *
196  * @param f the function
197  * @tparam I1 the type of the first parameter
198  * @tparam I2 the type of the second parameter
199 /**

```

The screenshot shows a Java IDE interface with two main panes. The left pane displays the project structure for 'CSYE7200 - Function.scala [FunctionalComposition]'. The right pane shows the content of the 'Function.scala' file.

**Project Tree:**

- Project
- CSYE7200-Spring2023 > assignment-functional-composition > src > main > scala > edu > neu > coe > csye7200 > asstfc > Function.scala
- src
  - main
    - scala
      - edu.neu.coe.csye7200.asstfc
        - CsvColumn
        - FC.sc
        - FoldLeft
        - Function
        - Ingest
        - lift\_etc.sc
        - Movie
  - test
    - resources
    - scala
      - edu.neu.coe.csye7200.asstfc
        - FunctionSpec
        - MovieSpec
- target
- build.sbt

**Function.scala Content:**

```
/*
 * @param R the result type of function f
 * @return a (curried) function of type (T1,T2)=>T4=>R
 */
// This one is a bit harder. But again, think in terms of an anonymous function that is what you want to return
def uncurried2[T1, T2, T3, R](f: T1 => T2 => T3 => R): (T1, T2) => T3 => R = ??? // TO BE IMPLEMENTED
(t1: T1, t2: T2) => (t3: T3) => f(t1)(t2)(t3)

/**
 * This method uncurries the first three parameters of a four- (or more-) parameter curried function.
 * The result is a (curried) function whose first parameter is a tuple of the first three parameters of f;
 * whose second parameter is the third parameter, etc.
 *
 * @param f the function
 * @param T1 the type of the first parameter
 * @param T2 the type of the second parameter
 * @param T3 the type of the third parameter
 * @param T4 the type of the fourth parameter
 * @param R the result type of function f
 * @return a (curried) function of type (T1,T2,T3)=>T4=>R
 */
// If you can do uncurried2, then you can do this one
def uncurried3[T1, T2, T3, T4, R](f: T1 => T2 => T3 => T4 => R): (T1, T2, T3) => T4 => R = ??? // TO BE IMPLEMENTED
(t1: T1, t2: T2, t3: T3) => (t4: T4) => f(t1)(t2)(t3)(t4)

/**
 * This method uncurries the first three parameters of a four- (or more-) parameter curried function.
 * The result is a (curried) function whose first parameter is a tuple of the first seven parameters of f;
 * whose second parameter is the third parameter, etc.
 *
 */
```

CSYE7200 - Function.scala [FunctionalComposition]

Project

CSYE7200-Spring2023 > assignment-functional-composition src main scala edu neu coe csye7200 asstfc ○ Function.scala

README.md × RandomState.scala × RandomStateSpec.scala × Ingest.scala × Function.scala × FunctionSpec.scala × Movie.scala ×

scalac -Xmax-classfile-name-length 1000

src

main

scala

edu.neu.coe.csye7200.asstfc

CsvColumn

FC.sc

FoldLeft

Function

Ingest

lift\_etc.sc

Movie

test

resources

scala

edu.neu.coe.csye7200.asstfc

FunctionSpec

MovieSpec

target

build.sbt

assignment-helloworld [HelloWorld]

assignment-lazy [Lazy]

assignment-movie-database [MovieDatabase]

assignment-random-state [RandomState]

src

main

225 \* This method uncurries the first three parameters of a four- (or more-) parameter curried function.

226 \* The result is a (curried) function whose first parameter is a tuple of the first seven parameters of f;

227 \* whose second parameter is the third parameter, etc.

228 \*

229 \* @param f the function

230 \* @tparam T1 the type of the first parameter

231 \* @tparam T2 the type of the second parameter

232 \* @tparam T3 the type of the third parameter

233 \* @tparam T4 the type of the fourth parameter

234 \* @tparam R the result type of function f

235 \* @return a (curried) function of type (T1,T2,T3)=>T4=>R

236 \*/

237 // If you can do uncurried3, then you can do this one

238 def uncurried7[T1, T2, T3, T4, T5, T6, T7, T8, R](f: T1 => T2 => T3 => T4 => T5 => T6 => T7 => T8 => R): (T1, T2, T3, T4, T5, T6, T7, T8) => R = uncurried3(f)(T1, T2, T3, T4)

239 //??? // TO BE IMPLEMENTED

240 (t1: T1, t2: T2, t3: T3, t4: T4, t5: T5, t6: T6, t7: T7) => (t8: T8) => f(t1)(t2)(t3)(t4)(t5)(t6)(t7)(t8)

241

242

243

244 def sequence[X](xs: Seq[Try[X]]): Try[Seq[X]] = xs.foldLeft(Try(Seq[X]())) {

245 (x, y) => for (x <- xs; y <- x) yield xs :+ x

246 }

247

## Movie.scala

```
CSYE7200 - Movie.scala [FunctionalComposition]
Project  RandomState.scala  RandomStateSpec.scala  Ingest.scala  Function.scala  FunctionSpec.scala  Movie.scala  MovieSpec.scala
src/main/edu.neu.coe.csye7200.asstfc
    CsvColumn
    FC.sc
    FoldLeft
    Function
    Ingest
    lift_etc.sc
    Movie
    test
        resources
        scala
            edu.neu.coe.csye7200.asstfc
                FunctionSpec
                MovieSpec
    target
    build.sbt
assignment-hello-world [HelloWorld]
assignment-lazy [Lazy]
assignment-movie-database [MovieDatabase]
assignment-random-state [RandomState]
```

```
//Hint: You may refer to the slides discussed in class for how to serialize object to json
object MoviesProtocol extends DefaultJsonProtocol {
    // 20 points
    // TO BE IMPLEMENTED

    implicit val formatSERFormat: RootJsonFormat[Format] = jsonFormat4(Format.apply)
    implicit val productionFormat: RootJsonFormat[Production] = jsonFormat4(Production.apply)
    implicit val ratingFormat: RootJsonFormat[Rating] = jsonFormat2(Rating.apply)
    implicit val reviewsFormat: RootJsonFormat[Reviews] = jsonFormat7(Reviews.apply)
    implicit val nameFormat: RootJsonFormat[Name] = jsonFormat4(Name.apply)
    implicit val principalFormat: RootJsonFormat[Principal] = jsonFormat2(Principal.apply)
    implicit val movieFormat: RootJsonFormat[Movie] = jsonFormat11(Movie.apply)

    // Define JSON protocol for MovieWithId case class
    //implicit val movieWithIdFormat = jsonFormat2(Movie.apply)
}
```

The screenshot shows a Java IDE interface with multiple tabs open. The main tab is titled "CSYE7200 - Movie.scala [FunctionalComposition]". Other tabs include "RandomState.scala", "RandomStateSpec.scala", "Ingest.scala", "Function.scala", "FunctionSpec.scala", "Movie.scala", "MovieSpec.scala", and "scala-at-light-speed - API.scala [scala-at-light-speed]".

The code in the "Movie.scala" tab is as follows:

```
137 //implicit val movieFormat: RootJsonFormat[Movie] = jsonFormat3(Movie)
138 //Hint: Serialize the input to Json format and deserialize back to Object, check the result is still equal to original
139 def testSerializationAndDeserialization(ms: Seq[Movie]): Boolean = {
140   // 5 points
141   // TO BE IMPLEMENTED
142
143   import MoviesProtocol._
144
145   val SerializeAndDeserialize = ms.map(_.toJson.convertTo[Movie])
146   ms == SerializeAndDeserialize
147 }
148
149
150 def getMoviesFromCountry(country: String, movies: Iterator[Try[Movie]]): Try[Seq[Movie]] = {
151   val movieSeq = Try(movies.map(_.get).toSeq)
152 }
```

### - Unit tests

## Functional

CSYE7200-Spring2023 > assignment-functional-composition

Project

src

- main
- scala
  - edu.neu.coe.csye7200.asstfc
    - CsvColumn
    - FC.sc
    - FoldLeft
    - Function
    - Ingest
    - lift\_etc.sc
    - Movie
- test
- resources
- docs

FunctionSpec

FunctionSpec.scala

1 package edu.neu.coe.csye7200.asstfc  
2  
3 import ..  
4  
5 class FunctionSpec extends AnyFlatSpec with Matchers {  
6  
7 behavior of "map2"  
8  
9 it should """"match Success(1234) for parse "12" to int and parse "34" to int, with (a:Int,b:Int) => a.toInt() + b.toInt()  
10 val a1 = "12"  
11 val a2 = "34"  
12 val t1 = Try(a1.toInt)  
13  
14 }

Run: FunctionSpec

Tests passed: 8 of 8 tests - 58ms

Test Results

Test	Time	Result
FunctionSpec	58ms	/Users/anirudhajoshi/Library/Java/JavaVirtualMachines/corretto-1.8.0_362/Contents/Home/bin/java ...
map2	32ms	✓
map7	9ms	✓
invert2	3ms	✓
invert3	7ms	✓
invert4	2ms	✓
uncurried2	5ms	✓
should work	5ms	✓

## Movie

The screenshot shows the IntelliJ IDEA interface with the following details:

- Project View:** Shows the file structure under "CSYE7200-Spring2023".
  - src: main/scala (containing CsvColumn, FC.sc, FoldLeft, Function, Ingest, lift\_etc.sc, Movie)
  - test: resources, scala (containing MovieSpec.scala)
- Code Editor:** Displays the content of `MovieSpec.scala`. The code defines a `MovieSpec` class extending `AnyFlatSpec with Matchers`. It includes imports and a block of code starting with `val phi: Double = (math.sqrt(5) + 1) / 2`.
- Run Tab:** Shows the results of a test run:
  - Tests passed: 16 of 16 tests – 373ms
  - Test Results tree:
    - MovieSpec (373ms)
      - Name (24 ms)
      - Principal (8 ms)
      - Rating (4 ms)
      - Format (9 ms)
      - Production (7 ms)
      - Reviews (6 ms)
    - Movie.getMoviesFromCo (201 ms)
      - should work for the sa (201 ms)
    - Movie.testsSerializationAn (114 ms)
      - should work for the sa (114 ms)
  - Message: "Testing started at 10:35 PM ..."