Anirudha Joshi – SEC01 (NUID 002991365)

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

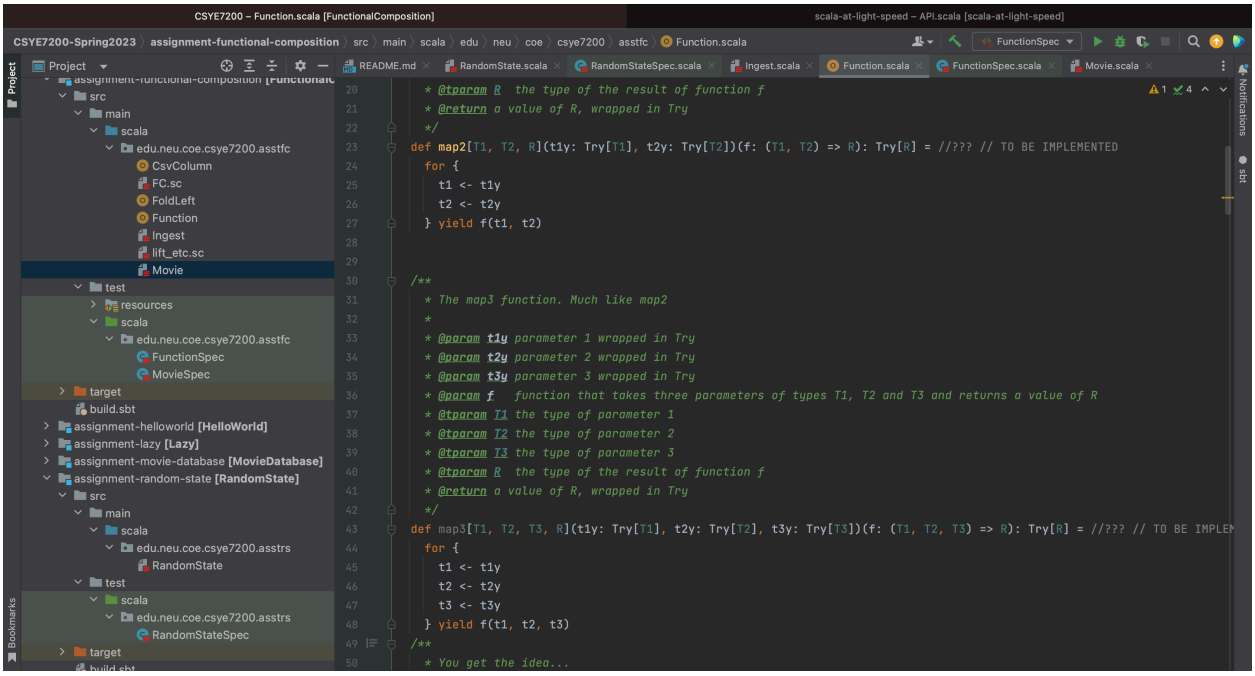
GitHub - https://github.com/anirudhajoshi2808/Anirudha_Joshi_CSYE7200

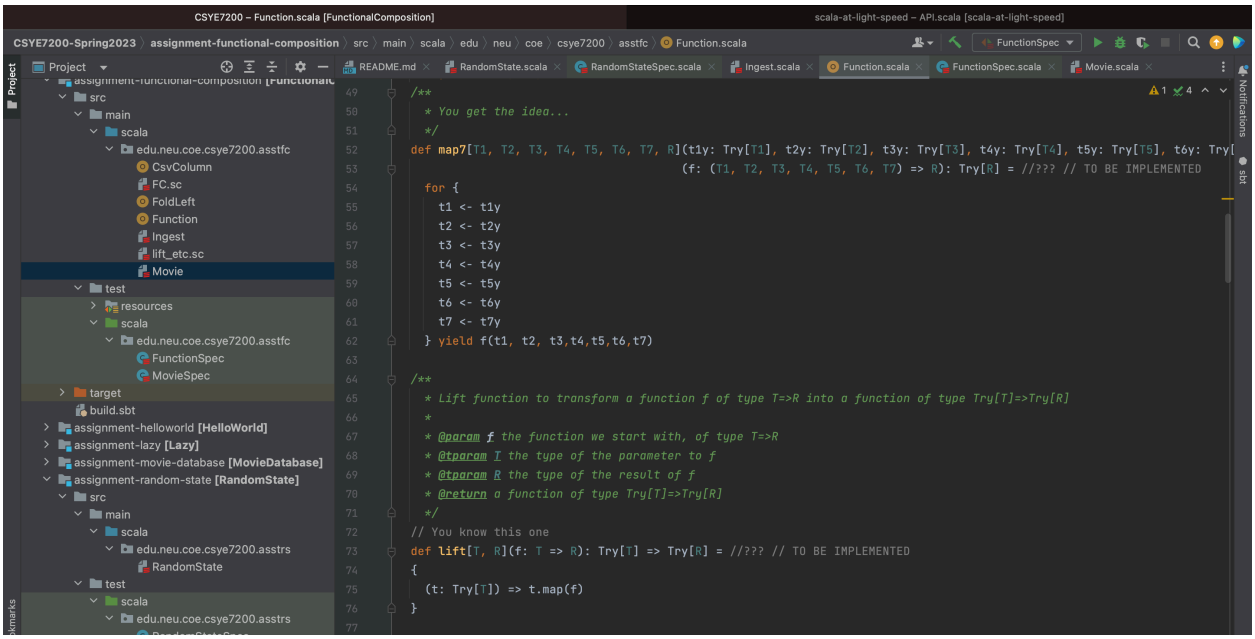
- Questions:

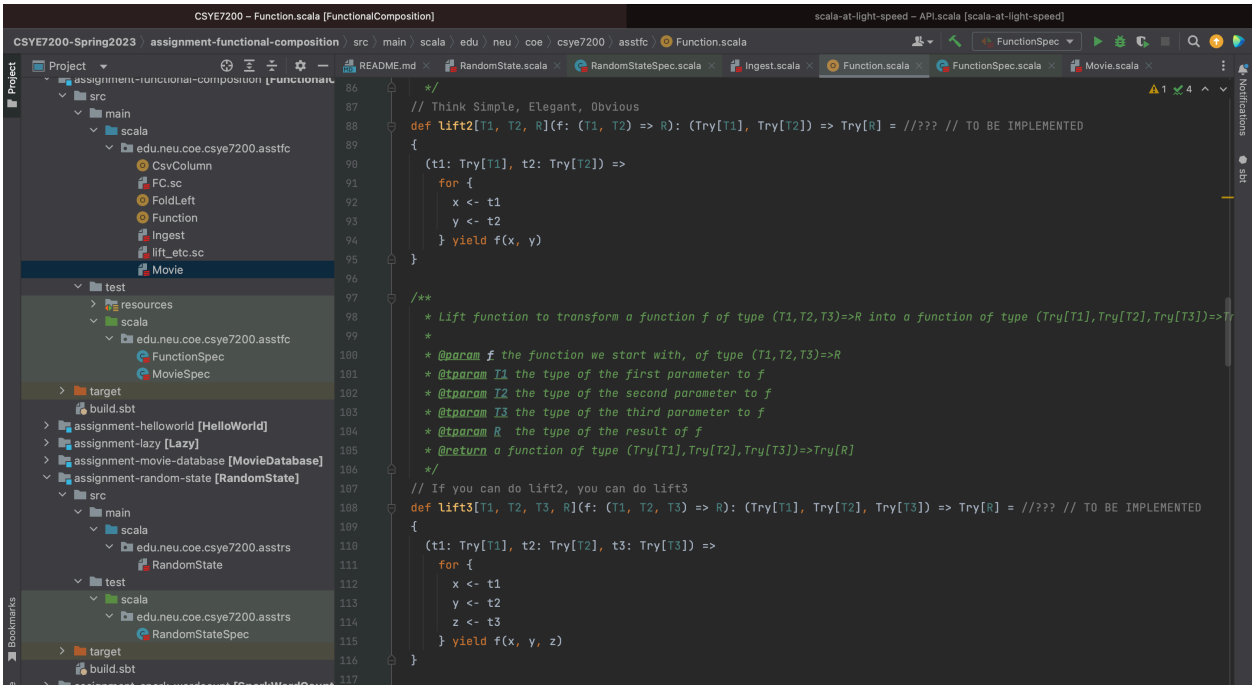
There are 13 TODOs in *Function.scala* and 2 TODOs in *Movie.scala*. You can get these from the class repo (see *Course Material/Resources/Class Repository*), the module name for this assignment is *assignment-functional-composition*.

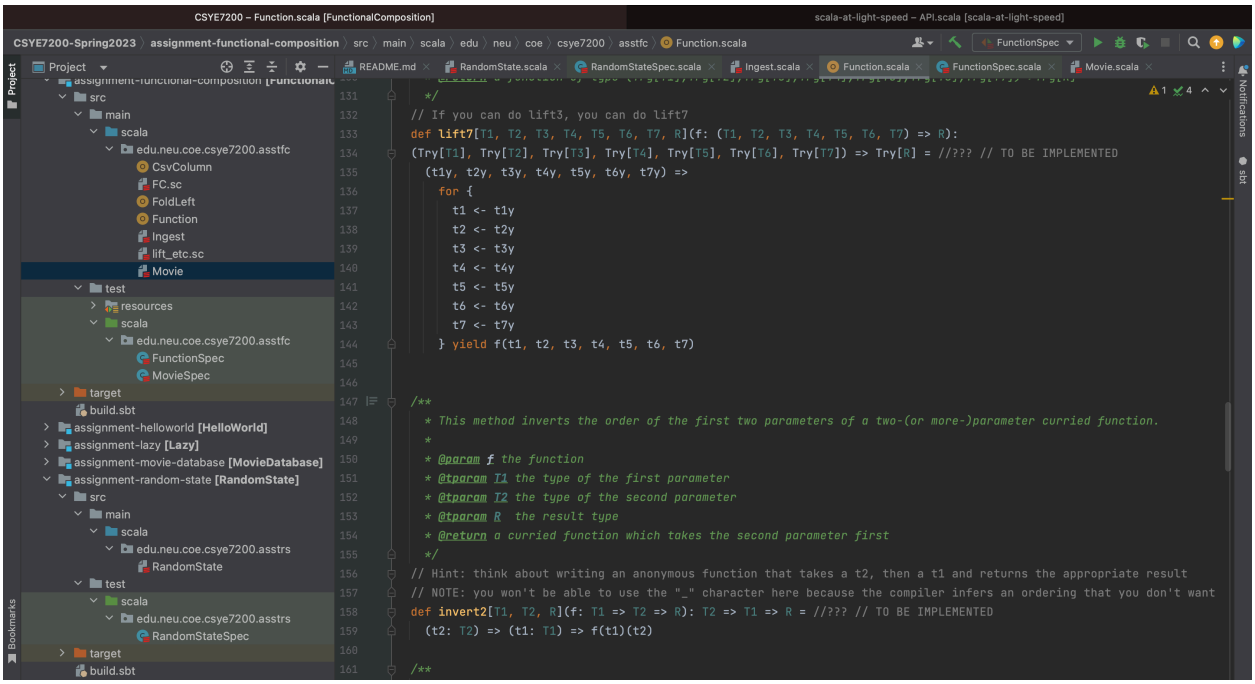
- Code

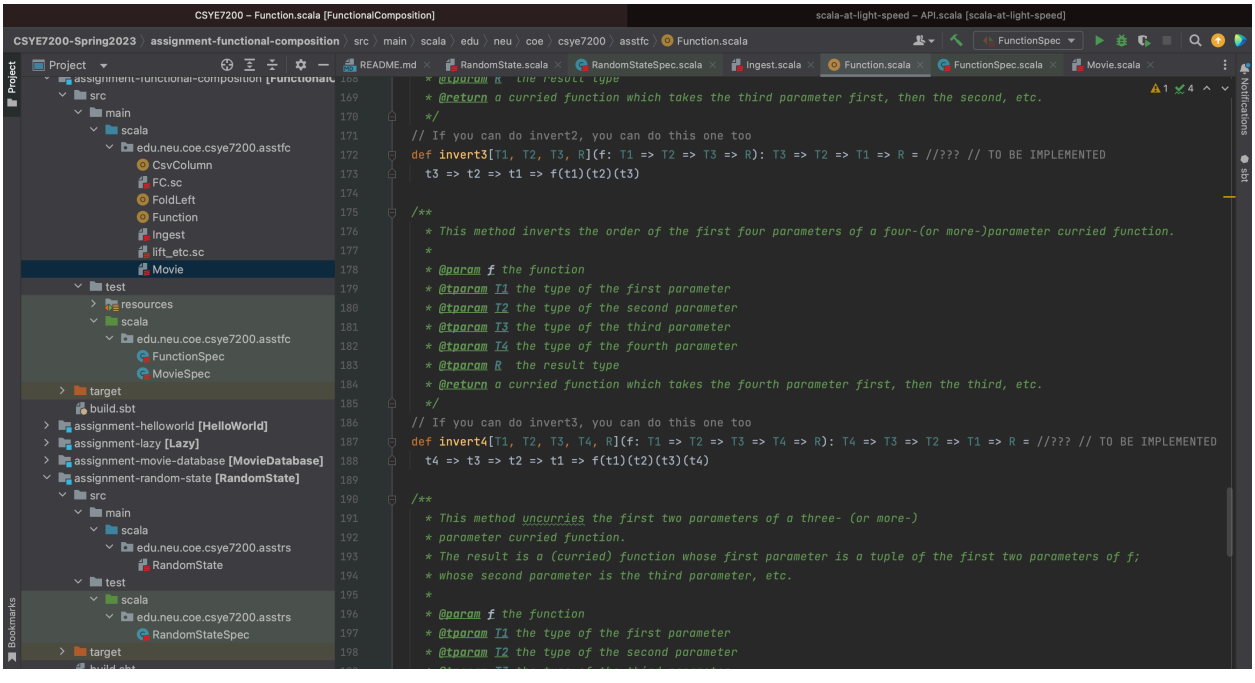
Function.scala

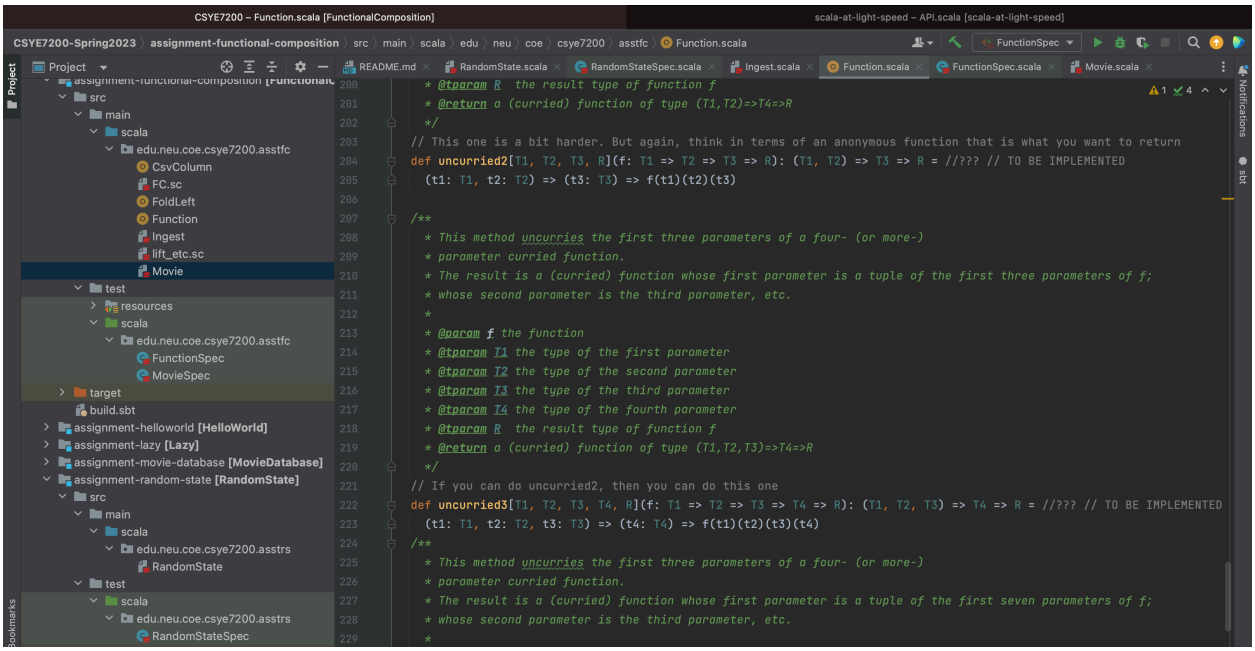


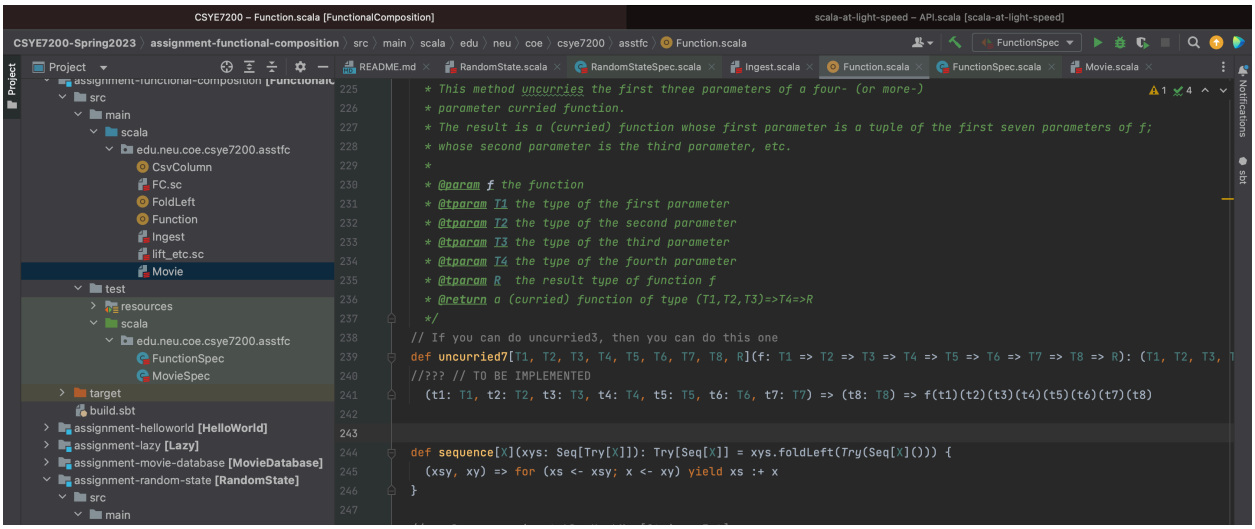

CSYE7200 – Function.scala [FunctionalComposition]
scala-at-light-speed – API.scala [scala-at-light-speed]
CSYE7200-Spring2023 assignment-functional-composition src main scala edu neu coe csye7200 asstfc Function.scala
FunctionSpec
Project
README.md
RandomState.scala
RandomStateSpec.scala
Ingest.scala
Function.scala
FunctionSpec.scala
Movie.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
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
Bookmarks
Notifications
sbt
/**
* You get the idea...
*/
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[
(f: (T1, T2, T3, T4, T5, T6, T7) => R): Try[R] = //??? // TO BE IMPLEMENTED
for {
t1 <- t1y
t2 <- t2y
t3 <- t3y
t4 <- t4y
t5 <- t5y
t6 <- t6y
t7 <- t7y
} yield f(t1, t2, t3,t4,t5,t6,t7)
/**
* Lift function to transform a function f of type T=>R into a function of type Try[T]=>Try[R]
*
* @param f the function we start with, of type T=>R
* @tparam T the type of the parameter to f
* @tparam R the type of the result of f
* @return a function of type Try[T]=>Try[R]
*/
// You know this one
def lift[T, R](f: T => R): Try[T] => Try[R] = //??? // TO BE IMPLEMENTED
{
(t: Try[T]) => t.map(f)
}


CSYE7200 – Function.scala [FunctionalComposition]
scala-at-light-speed – API.scala [scala-at-light-speed]
CSYE7200-Spring2023 assignment-functional-composition src main scala edu neu coe csye7200 asstfc Function.scala
FunctionSpec
Project
README.md
RandomState.scala
RandomStateSpec.scala
Ingest.scala
Function.scala
FunctionSpec.scala
Movie.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
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
Bookmarks
Notifications
sbt
*/
// Think Simple, Elegant, Obvious
def lift2[T1, T2, R](f: (T1, T2) => R): (Try[T1], Try[T2]) => Try[R] = //??? // TO BE IMPLEMENTED
{
(t1: Try[T1], t2: Try[T2]) =>
for {
x <- t1
y <- t2
} yield f(x, y)
}
/**
* Lift function to transform a function f of type (T1,T2,T3)=>R into a function of type (Try[T1],Try[T2],Try[T3])=>Tr
*
* @param f the function we start with, of type (T1,T2,T3)=>R
* @tparam T1 the type of the first parameter to f
* @tparam T2 the type of the second parameter to f
* @tparam T3 the type of the third parameter to f
* @tparam R the type of the result of f
* @return a function of type (Try[T1],Try[T2],Try[T3])=>Try[R]
*/
// If you can do lift2, you can do lift3
def lift3[T1, T2, T3, R](f: (T1, T2, T3) => R): (Try[T1], Try[T2], Try[T3]) => Try[R] = //??? // TO BE IMPLEMENTED
{
(t1: Try[T1], t2: Try[T2], t3: Try[T3]) =>
for {
x <- t1
y <- t2
z <- t3
} yield f(x, y, z)
}

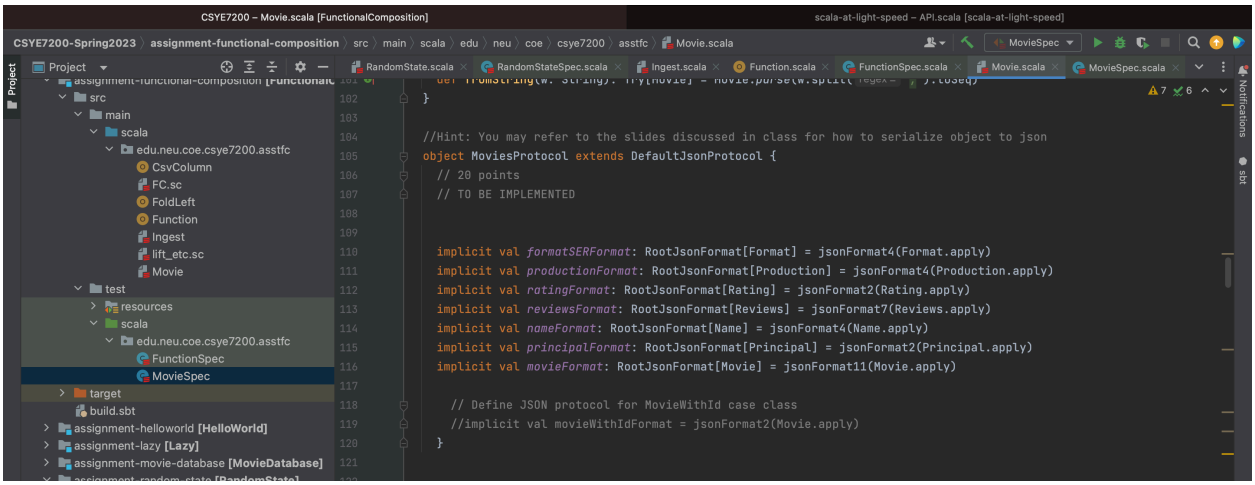

CSYE7200 – Function.scala [FunctionalComposition]
scala-at-light-speed – API.scala [scala-at-light-speed]
CSYE7200-Spring2023 assignment-functional-composition src main scala edu neu coe csye7200 asstfc Function.scala
FunctionSpec
Project
README.md
RandomState.scala
RandomStateSpec.scala
Ingest.scala
Function.scala
FunctionSpec.scala
Movie.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
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
*/
// If you can do lift3, you can do lift7
def lift7[T1, T2, T3, T4, T5, T6, T7, R](f: (T1, T2, T3, T4, T5, T6, T7) => R):
(Try[T1], Try[T2], Try[T3], Try[T4], Try[T5], Try[T6], Try[T7]) => Try[R] = //??? // TO BE IMPLEMENTED
(t1y, t2y, t3y, t4y, t5y, t6y, t7y) =>
for {
t1 <- t1y
t2 <- t2y
t3 <- t3y
t4 <- t4y
t5 <- t5y
t6 <- t6y
t7 <- t7y
} yield f(t1, t2, t3, t4, t5, t6, t7)
/**
* This method inverts the order of the first two parameters of a two-(or more-)parameter curried function.
*
* @param f the function
* @tparam T1 the type of the first parameter
* @tparam T2 the type of the second parameter
* @tparam R the result type
* @return a curried function which takes the second parameter first
*/
// Hint: think about writing an anonymous function that takes a t2, then a t1 and returns the appropriate result
// NOTE: you won't be able to use the "_" character here because the compiler infers an ordering that you don't want
def invert2[T1, T2, R](f: T1 => T2 => R): T2 => T1 => R = //??? // TO BE IMPLEMENTED
(t2: T2) => (t1: T1) => f(t1)(t2)
/**

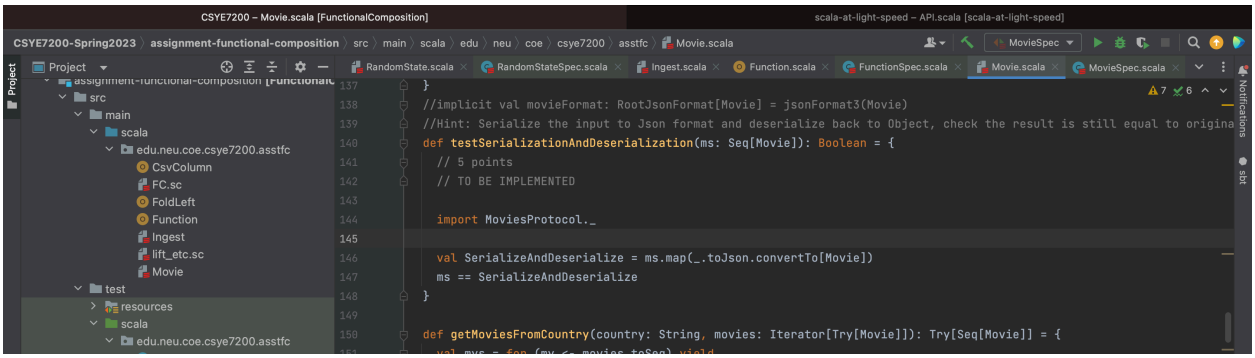

CSYE7200 – Function.scala [FunctionalComposition]
scala-at-light-speed – API.scala [scala-at-light-speed]
CSYE7200-Spring2023 assignment-functional-composition src main scala edu neu coe csye7200 asstfc Function.scala
FunctionSpec
Project
README.md
RandomState.scala
RandomStateSpec.scala
Ingest.scala
Function.scala
FunctionSpec.scala
Movie.scala
* @return a curried function which takes the third parameter first, then the second, etc.
*/
// If you can do invert2, you can do this one too
def invert3[T1, T2, T3, R](f: T1 => T2 => T3 => R): T3 => T2 => T1 => R = //??? // TO BE IMPLEMENTED
t3 => t2 => t1 => f(t1)(t2)(t3)
/**
* This method inverts the order of the first four parameters of a four-(or more-)parameter curried function.
*
* @param f the function
* @tparam T1 the type of the first parameter
* @tparam T2 the type of the second parameter
* @tparam T3 the type of the third parameter
* @tparam T4 the type of the fourth parameter
* @tparam R the result type
* @return a curried function which takes the fourth parameter first, then the third, etc.
*/
// If you can do invert3, you can do this one too
def invert4[T1, T2, T3, T4, R](f: T1 => T2 => T3 => T4 => R): T4 => T3 => T2 => T1 => R = //??? // TO BE IMPLEMENTED
t4 => t3 => t2 => t1 => f(t1)(t2)(t3)(t4)
/**
* This method uncurries the first two parameters of a three- (or more-)
* parameter curried function.
* The result is a (curried) function whose first parameter is a tuple of the first two parameters of f;
* whose second parameter is the third parameter, etc.
*
* @param f the function
* @tparam T1 the type of the first parameter
* @tparam T2 the type of the second parameter


CSYE7200 – Function.scala [FunctionalComposition]
scala-at-light-speed – API.scala [scala-at-light-speed]
CSYE7200-Spring2023 assignment-functional-composition src main scala edu neu coe csye7200 asstfc Function.scala
FunctionSpec
Project
README.md
RandomState.scala
RandomStateSpec.scala
Ingest.scala
Function.scala
FunctionSpec.scala
Movie.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
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
* @tparam 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
* @tparam T1 the type of the first parameter
* @tparam T2 the type of the second parameter
* @tparam T3 the type of the third parameter
* @tparam T4 the type of the fourth parameter
* @tparam 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]
scala-at-light-speed – API.scala [scala-at-light-speed]
CSYE7200-Spring2023 assignment-functional-composition src main scala edu neu coe csye7200 asstfc Function.scala
FunctionSpec
* 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.
*
* @param f the function
* @tparam T1 the type of the first parameter
* @tparam T2 the type of the second parameter
* @tparam T3 the type of the third parameter
* @tparam T4 the type of the fourth parameter
* @tparam R the result type of function f
* @return a (curried) function of type (T1,T2,T3)=>T4=>R
*/
// If you can do uncurried3, then you can do this one
def uncurried7[T1, T2, T3, T4, T5, T6, T7, T8, R](f: T1 => T2 => T3 => T4 => T5 => T6 => T7 => T8 => R): (T1, T2, T3,
//??? // TO BE IMPLEMENTED
(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)
def sequence[X](xys: Seq[Try[X]]): Try[Seq[X]] = xys.foldLeft(Try(Seq[X]())) {
(xsy, xy) => for (xs <- xsy; x <- xy) yield xs :+ x
}

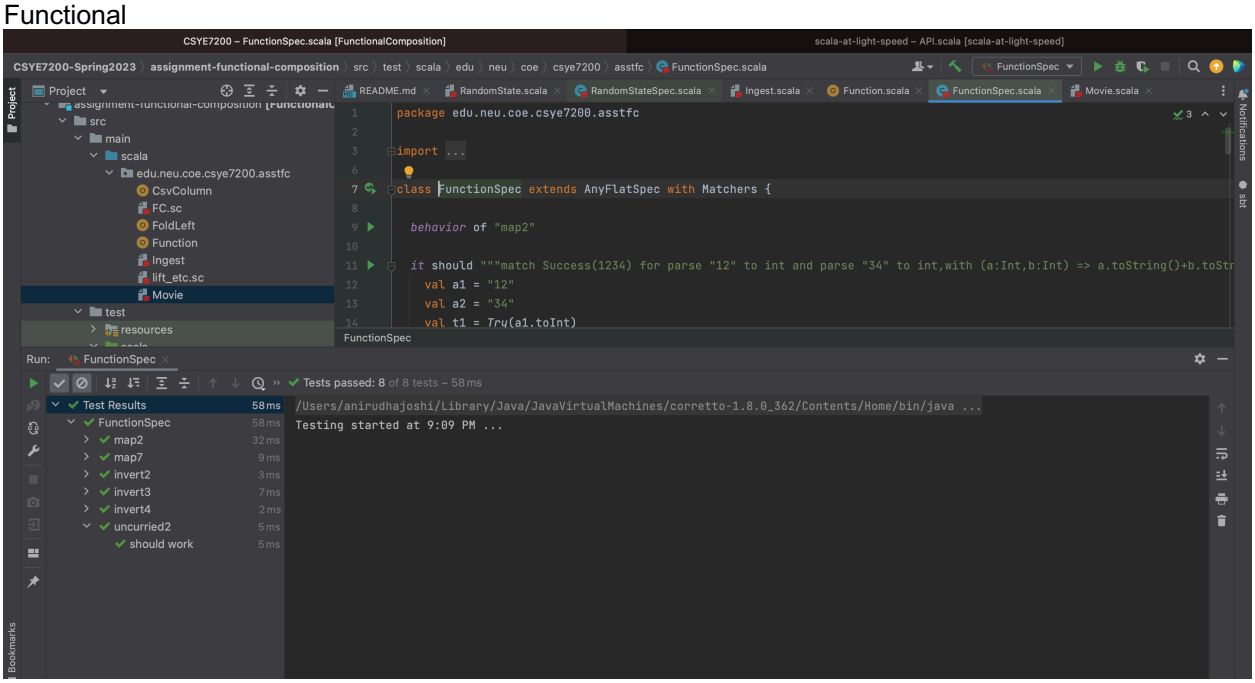
Movie.scala



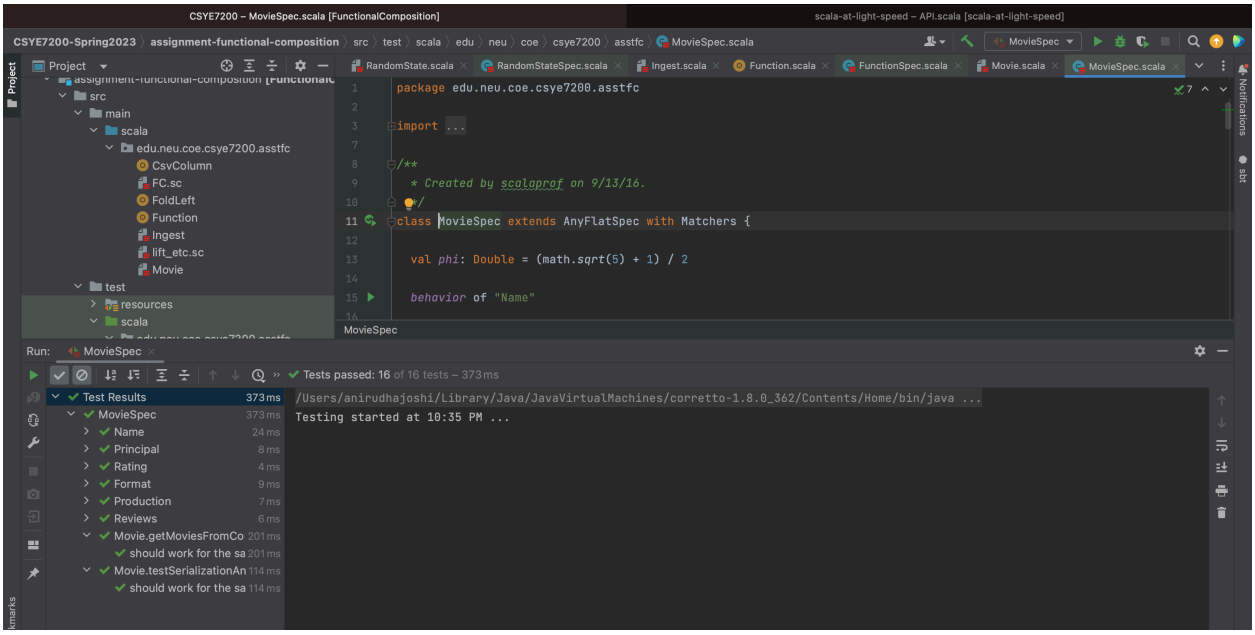


- Unit tests

Functional



Movie


CSYE7200 – MovieSpec.scala [FunctionalComposition]
scala-at-light-speed – API.scala [scala-at-light-speed]
CSYE7200-Spring2023 〉 assignment-functional-composition 〉 src 〉 test 〉 scala 〉 edu 〉 neu 〉 coe 〉 csye7200 〉 asstfc 〉 MovieSpec.scala
Project
src
main
scala
edu.neu.coe.csye7200.asstfc
CsvColumn
FC.sc
FoldLeft
Function
Ingest
lift_etc.sc
Movie
test
resources
scala
RandomState.scala
RandomStateSpec.scala
Ingest.scala
Function.scala
FunctionSpec.scala
Movie.scala
MovieSpec.scala
package edu.neu.coe.csye7200.asstfc
import ...
/**
* Created by scalaprof on 9/13/16.
*/
class MovieSpec extends AnyFlatSpec with Matchers {
val phi: Double = (math.sqrt(5) + 1) / 2
behavior of "Name"
MovieSpec
Run:
MovieSpec
Tests passed: 16 of 16 tests – 373 ms
Test Results 373 ms
MovieSpec 373 ms
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.testSerializationAn 114 ms
should work for the sa 114 ms
/Users/anirudhajoshi/Library/Java/JavaVirtualMachines/corretto-1.8.0_362/Contents/Home/bin/java ...
Testing started at 10:35 PM ...