Yue Chen – SEC01 (NUID 001044461)

Big Data System Engineering with Scala
Spring 2022
Assignment No.5

My implementations

Function.scala - 13 todos:

```
Yue Chen
23  def map2[T1, T2, R](t1y: Try[T1], t2y: Try[T2])(f: (T1, T2) => R): Try[R] = for(t1 <- t1y; t2<-t2y) yield f(t1,t2) //
```

```
Yue Chen
39  def map3[T1, T2, T3, R](t1y: Try[T1], t2y: Try[T2], t3y: Try[T3])(f: (T1, T2, T3) => R): Try[R] = for(t1<- t1y;t2<-t2y; t3<-t3y) yield f(t1, t2, t3)//
```

```
Yue Chen +1
44  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], t7y: Try[T7])
45                                          (f: (T1, T2, T3, T4, T5, T6, T7) => R): Try[R] =
46    for(t1 <- t1y;t2 <- t2y; t3 <- t3y; t4 <- t4y; t5 <- t5y; t6 <- t6y; t7 <- t7y) yield f(t1,t2,t3,t4,t5,t6,t7)// TO BE IMPLEMENTED
```

```
Yue Chen
57  def lift[T, R](f: T => R): Try[T] => Try[R] = _ map f //
```

```
Yue Chen
69  def lift2[T1, T2, R](f: (T1, T2) => R): (Try[T1], Try[T2]) => Try[R] = map2(_, _)(f) //
```

```
Yue Chen
82  def lift3[T1, T2, T3, R](f: (T1, T2, T3) => R): (Try[T1], Try[T2], Try[T3]) => Try[R] = map3(_, _, _)(f) //
```

```
99  def lift7[T1, T2, T3, T4, T5, T6, T7, R](f: (T1, T2, T3, T4, T5, T6, T7) => R):
100 (Try[T1], Try[T2], Try[T3], Try[T4], Try[T5], Try[T6], Try[T7]) => Try[R] = map7(_,_,_,_,_,_,_)(f) //
```

```
Yue Chen
113 def invert2[T1, T2, R](f: T1 => T2 => R): T2 => T1 => R = T2 => T1 => f(T1)(T2) //
```

```
125 // If you can do invert2, you can do this one too
    Yue Chen
126 def invert3[T1, T2, T3, R](f: T1 => T2 => T3 => R): T3 => T2 => T1 => R = T3 => T2 => T1 => f(T1)(T2)(T3) //
```

```
Yue Chen
140 def invert4[T1, T2, T3, T4, R](f: T1 => T2 => T3 => T4 => R): T4 => T3 => T2 => T1 => R = T4 => T3 => T2 => T1 => f(T1)(T2)(T3)(T4) //
```

```
Yue Chen
156 def uncurried2[T1, T2, T3, R](f: T1 => T2 => T3 => R): (T1, T2) => T3 => R = (t1:T1,t2: T2) => f(t1)(t2) //
```

```
Yue Chen
173 def uncurried3[T1, T2, T3, T4, R](f: T1 => T2 => T3 => T4 => R): (T1, T2, T3) => T4 => R = (t1:T1,t2: T2,t3:T3) => f(t1)(t2)(t3) //
```

```
190 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 =
191 (t1:T1,t2: T2,t3:T3, t4:T4, t5:T5, t6:T6, t7:T7) => f(t1)(t2)(t3)(t4)(t5)(t6)(t7) // TO BE IMPLEMENTED
```

Movie.scala - 2 todos:

```
Yue Chen +1
object MoviesProtocol extends DefaultJsonProtocol {
  // 20 points
  // TO BE IMPLEMENTED
  implicit val formatFormat: RootJsonFormat[Format] = jsonFormat4(Format.apply)
  implicit val nameFormat: RootJsonFormat[Name] = jsonFormat4(Name.apply)
  implicit val ratingFormat: RootJsonFormat[Rating] = jsonFormat2(Rating.apply)
  implicit val reviewFormat: RootJsonFormat[Reviews] = jsonFormat7(Reviews.apply)
  implicit val productionFormat: RootJsonFormat[Production] = jsonFormat4(Production.apply)
  implicit val principalFormat: RootJsonFormat[Principal] = jsonFormat2(Principal.apply)
  implicit val movieJsonFormat: RootJsonFormat[Movie] = jsonFormat11(Movie.apply)
}
```

```
Yue Chen +1
def testSerializationAndDeserialization(ms: Seq[Movie]): Boolean = {
  // 5 points
  // TO BE IMPLEMENTED
  import MoviesProtocol._
  for (m <- ms) {
    val json = m.toJson
    val movies = json.convertTo[Movie]
    if (!m.equals(movies)) false
  }
  true
}
```

All unit test passed