Unit 5e: Summary

1 Summary

A range of common functional programming concepts have been demonstrated using Scala. Other languages implement some or all of these concepts in their own way.

- **First class functions**: functions can be used in the same way as variables
- **Nested functions:** functions can be defined inside other functions
- **Optimising recursion:** recursion can cause stack memory problems if not optimised for tail recursion, Scala compiler supports this
- **Partial function application:** can apply a function to only some of its parameters, returns function that can be used elsewhere
- **Higher-order (HO) functions:** functions that take other functions as parameters or return other functions
- **Composition:** flexible creation of complex functionality by composing functions, typically by passing a function as a parameter to a HO function, widely used and supported when working with collections
- **Closures:** functions are closed over variables that are in scope where function declared, useful for communicating information between scopes when composing
- **Currying:** this is mapping from a single list with multiple parameters to multiple lists each with a single parameter, creates a function chain, can help write clear code when calling functions
- **Folding:** combines elements of a collection into a single value, using a function that defines how adjacent elements are combined
- **Streams:** allow lazy evaluation of large and infinite sequences, computationally efficient when working with parts of infinite sequences

2 Additional reading

https://spring.io/blog/2014/03/20/manning-publications-first-class-functions-in-java-8

http://alvinalexander.com/scala/how-to-use-partially-applied-functions-in-scala-syntax-examples

http://ananthakumaran.in/2010/03/29/scala-underscore-magic.html