Ubaid Ur Rehman

BSCS-5B

Reg. number: 00000123495

uurehman.bscs15seecs@seecs.edu.pk

Advanced Programming

Lab 4 Document (Functional Programming Set in Scala)

# Introduction

# In this open ended lab the students have to use functional paradigm in Scala and provide solution to the problem set.

# Objectives

# • Familiarize yourself with the use of Functional Programming Paradigm in Scala.

# • Write tests to show your solutions are correct.

# Approach

Simple Scala code is been written in task.sc file which is interpreted automatically when saved.

# Output:

import util.Random.nextInt

object Tasks {

/\*generating a list of 10 numbers with the range from 0-20 (for task 1 & 2)\*/

val myList = List.fill(10)(20).map(scala.util.Random.nextInt)

/\*1. \*Find out the last Nth element in a randomly generated list.\*/

def lastNth (num : Int, integersList : List[Int]) : Int = integersList(integersList.length - num)

lastNth (3, myList)

/\*2. \*Find out the last Nth element in a randomly generated list using a tail recursive

solution. \*/

def lastNthRecursive (num : Int, integersList : List[Int]) : Int = {

def myFunc(i : Int, index : Int, subList : List[Int]) : Int = {

if (i == index) subList.head

else myFunc(i + 1, index, subList.tail)

}

if (num >= integersList.length) 0

else myFunc(0, integersList.length - num, integersList)

}

lastNthRecursive (3, myList)

/\*3. \*Determine whether a given integer number is prime or not\*/

def isPrime(num : Int) : Boolean = {

if(num <= 1) false

else if (num == 2) true

else !(2 to (num - 1)).exists(x => num % x == 0)

}

isPrime(8)

isPrime(7)

/\*4. \*\*Determine whether two positive integer numbers are coprime.( Two numbers are coprime if

their greatest common divisor equals 1)\*/

def isCoPrime(num1 : Int, num2 : Int) : Boolean = {

var count = 0; //counter for loop

for(count <- 2 until (num1)){

if(num1 % count == 0 && num2 % count == 0) false

}

true

}

isCoPrime(9, 5)

isCoPrime(5, 10)

/\*5. \*\*Define XOR operation for two logical expressions.\*/

def oprXOR(x : Boolean, y : Boolean) : Boolean = {

if (x == y) false

else true

}

oprXOR(true,true)

oprXOR(false,true)

oprXOR(true,false)

oprXOR(false,false)

/\*6. \*\*Define AND operation for two logical expressions\*/

def oprAND(x : Boolean, y : Boolean) : Boolean = {

if (x == true && y == true) true

else false

}

oprAND(true,true)

oprAND(false,true)

oprAND(true,false)

oprAND(false,false)

/\*7. \*Define NAND operation for two logical expressions.\*/

def oprNAND(x : Boolean, y : Boolean) : Boolean = {

if (x == true && y == true) false

else true

}

oprNAND(true,true)

oprNAND(false,true)

oprNAND(true,false)

oprNAND(false,false)

/\*8. \*\*\*Given 5 random numbers (between 1 and 100), determine the mathematical operations

(sum, subtract, product, divide) which can produce a number X, where each number is used only

once. Example: How can 3, 7, 6, 8, 1 produce 348? (((8 \* 7) + 3) -1) \*6 = 348\*/

def howCanProduce(n1 : Int, n2 : Int, n3 : Int, n4 : Int, n5 : Int, result : Int) : Unit = {

}

}

Github link: https://github.com/uurehman/ScalaFunctionalProblemSet