**Group Activity 11; CS 3060**

**Names** of students in your group: Nicholas W, Nicholas B, Dylan S, Ethan F, Tyson S

Points: 10

**Goal**: To make us familiar with Scala’s support for map-reduce, fold, and parallel computing functionality.

**Task 1**: (4 points)

1. Write a function *foo1* which takes a list of integers x and computes (using *map*) another list y where i-th element of y is equal to 2^(i-th element of x). Then, *foo1* computes the sum of list y and returns the sum. Use *map-reduce* style. As an example, if the input is List(3, 5, 8), then *foo1* returns the value of (2^3 + 2^5 + 2^8). Avoid using any *var* in your code.
2. Now let’s utilize parallel computation to do the above job. Test with a random list (e.g. with 10 random small integers or so) to check whether the parallel computation saves time compared to the serial one.

**Task 2**: (3 points) Write a function *foo2* which takes a list of strings. Function *foo2* concatenates the first character of the input strings together with a ‘hyphen’ character as a separator, and returns the resultant string. As an example, if the input is List(“12ab”, “cde”, “xyz”), then *foo2* returns “1-c-x”. Use *map*-*reduce* style to do the job. And, try to avoid using any *var* in your code.

**Task 3**: (3 points) Write a function *foo3* which takes a list of strings. Function *foo3* concatenates the first character of the input strings together with a ‘hyphen’ character as a separator, and returns the resultant string. As an example, if the input is List(“12ab”, “cde”, “xyz”), then *foo3* returns “1-c-x”. Use *map* and *fold* (instead of *reduce*) to do the job. And, try to avoid using any *var* in your code.

Hint: We have discussed *foldLeft* in ppt and *fold* is similar. You may also check the following link.

<https://coderwall.com/p/4l73-a/scala-fold-foldleft-and-foldright>

Task 1:

// import

import scala.math.pow

import scala.io.\_

def foo1(*x*: List[Int]) = {

    val y = x.map(num => pow(2, num))

    val sum = y.reduce(\_+\_)

    println(y)

    println(sum)

}

def foo1Par(*x*: List[Int]) = {

    val y = x.par map(num => pow(2, num))

    val sum = y.reduce(\_+\_)

    println(y)

    println(sum)

}

val t1 = System.currentTimeMillis()

foo1(List(2, 5, 8, 3, 6, 8, 3, 6, 2, 7))

val t2 = System.currentTimeMillis()

val t3 = System.currentTimeMillis()

foo1Par(List(2, 5, 8, 3, 6, 8, 3, 6, 2, 7))

val t4 = System.currentTimeMillis()

println("serial exec time = " + (t2 - t1))

println("par exec time = " + (t4 - t3))

Task 2:

// import

def foo2(*stringList*: List[String]) = {

    //val newList = stringList.map(str =>

    //    if(str == stringList(stringList.size - 1)){

    //        str.charAt(0).toString

    //    }else{

    //        str.charAt(0).toString + "-"

    //    }

    //).reduce(\_+\_)

    val newList = stringList.map(str => str.charAt(0).toString)

    val result = newList.reduce(\_+"-"+\_)

    println(result)

}

foo2(List("12ab", "cde", "xyz"))

Task 3:

// import

def foo3(*stringList*: List[String]) = {

    val newList = stringList.map(str =>

        if(str == stringList(stringList.size - 1))

        {

            str.charAt(0).toString

        }else{

            str.charAt(0).toString + "-"

        }

    ).fold(""){(sum,item) => sum+item}

    println(result)

}

foo3(List("12ab", "cde", "xyz"))