**CSC 369 – Assignment 6**

**Question 1:**

object TakenDifficult {

def findMax(courses:List[String]) : Int = {

val max = -1;

var temp = 0;

for (line <- courses) {

temp = Integer.parseInt(line.split(",")(1))

if (temp > max)

max = temp

}

return max;

}

def main(args:Array[String]) : Unit = {

val students = sc.textFile("../students.txt")

val courses = sc.textFile("../courses.txt")

val grades = sc.textFile("../grades.txt")

val studentMap = students.map(line => (line.split(",")(0).toInt,

(line.split(",")(1), (line.split(",")(2),

line.split(",")(3) ))))

val courseMap = courses.map(line => (line.split(",")(0)))

val gradeMap = grades.map(line => (line.split(",")))

**// Order students: (id, name)**

val idStudent = students.map {case (id, (name, (address, num)))

=> (id, name)

}

**// (id, (student, (course, grade)))**

val stuGrades = idStudent.join(gradeMap);

val courseStuGrade = stuGrades.map {

case (id, (name, (course, grade))) =>

(course, (name, grade))

}

**// This yields: (course, ((name, grade), difficulty))**

val courseStuGradeDiff = courseStuGrade.join(courseMap)

**// This yeilds: (name, difficulty)**

val studentDiff = courseStuGradeDiff.map {

case (course, ((name, grade), diff)) => (name, diff)

}

**// Print the students who have taken the most difficult course**

val maxDiff = findMax(courses)

for (line <- studentDiff){

if (line.contains(maxDiff)){

println(line(0))

}

}

}

}

**Question 2:**

object AvgDiff {

def main(args:Array[String]) : Unit = {

val students = sc.textFile("../students.txt")

val courses = sc.textFile("../courses.txt")

val grades = sc.textFile("../grades.txt")

val studentMap = students.map(line => (line.split(",")(0).toInt,

(line.split(",")(1), (line.split(",")(2),

line.split(",")(3) ))))

val courseMap = courses.map(line => (line.split(",")(0)))

val gradeMap = grades.map(line => (line.split(",")))

**// Order students: (id, name)**

val idStudent = students.map {case (id, (name, (address, num)))

=> (id, name)}

**// (id, (student, (course, grade)))**

val stuGrades = idStudent.join(gradeMap);

val courseStuGrade = stuGrades.map {

case (id, (name, (course, grade))) =>

(course, (name, grade))

}

**// This yields: (course, ((name, grade), difficulty))**

val courseStuGradeDiff = courseStuGrade.join(courseMap)

**// This yeilds: (name, difficulty)**

val studentDiff = courseStuGradeDiff.map {

case (course, ((name, grade), diff)) =>

(name, diff.toInt)

}

val combinedStudentDiff = studentDiff.combineByKey(

(score) => (score, 1),

(acc: (Int, Int), score) => (acc.\_1 + score, acc.\_2 + 1),

(acc1: (Int, Int), acc2(Int, Int)) =>

(acc1.\_1 + acc2.\_1, acc1.\_2 + acc2.\_2)).map({

case (key, value) =>

(key, value.\_1 \*1.0 / value.\_2)

})

)

**// Print each entry**

combinedStudentDiff.foreach(line =>

println(line.split(",")(0) + line.split(",")(1)))

}

}

**Question 3:**

object TopDifficult {

def main(args:Array[String]) : Unit = {

val courses = sc.textFile("../courses.txt")

val courseMap = courses.map(line => (line.split(",")(0)))

val courseDiff = courseMap.map{

case (course, diff) => (diff, course)

}.sortByKey(false) **// sort descending**

courseDiff.take(5).foreach(tuple => println(tuple.\_1 +

" " + tuple.\_2))

}

}

**Question 4:**

object TopGPA {

def main(args:Array[String]) : Unit = {

val gradeMap = Map("A" -> 4, "B" -> 3, "C" -> 2, "D" -> 1,

"F" -> 0)

val students = sc.textFile("../students.txt")

val grades = sc.textFile("../grades.txt")

val studentMap = students.map(line => (line.split(",")(0).toInt,

(line.split(",")(1), (line.split(",")(2),

line.split(",")(3) ))))

val gradesMap = grades.map(line => (line.split(",")))

**// Order students: (id, name)**

val idStudent = students.map {case (id, (name, (address, num)))

=> (id, name)}

**// (id, (student, (course, grade)))**

val studentJoin = idStudent.join(gradeMap);

// Order: (name, grade-point)

val stuGrades = studentJoin.map {

case (id, (name, (course, grade))) =>

(name, gradeMap(grade))

}

val combinedStudentGpa = stuGrades.combineByKey(

(score) => (score, 1),

(acc: (Int, Int), score) => (acc.\_1 + score, acc.\_2 + 1),

(acc1: (Int, Int), acc2(Int, Int)) =>

(acc1.\_1 + acc2.\_1, acc1.\_2 + acc2.\_2)).map({

case (key, value) =>

(key, value.\_1 \*1.0 / value.\_2)

})

)

**// Print each entry**

combinedStudentDiff.foreach(line =>

println(line.\_1 + " " + line.\_2))

}

}