Name: Winton Gee

Q1[10]. The program reads a file full of integers and computes the number of times each integer that is divisible by 3 occurs.

```
 def firstPart(): Unit = {
   val conf = new SparkConf().setAppName("AppName").setMaster("local")
   val sc = new SparkContext(conf)
   val firstInputRdd = sc.textFile(inputPath_1).flatMap(_.split(" "))
   firstInputRdd
     .map(_.toInt) // Convert the string input into int type
     .filter(num => num % 3 == 0) // Only count divisible by 3
     .groupBy(identity) // Used to group the values by original values
     .mapValues { numList => // Sum number of times each number occurs
       numList.size // Instead of sum, do size because that is the number of elements in that list
of same numbers
     }
     .foreach {
       case (num, count) =>
         println(num + " appears " + count + " times")
     }
 }
```

Q2[10]. The program reads a file with employees and a file with departments. The program should print the employee name and department name for each employee.

```
 def secondPart(): Unit = {
   val conf = new SparkConf().setAppName("AppName").setMaster("local")
   val sc = new SparkContext(conf)
   val employeeRdd = sc.textFile(employeePath).map(_.split(", "))
   val departmentRdd = sc.textFile(departmentPath).map(_.split(", "))

   employeeRdd
     .cartesian(departmentRdd) // Join
     .filter {
       case (employee, department) => // Make sure the ID's match
         employee(1).equals(department(0))
     }
     .map {
       case (employee, department) =>
         employee(0) + ", " + department(1)
     }
     .foreach(println)
 }
```

Q3[10]. Write program that prints the student name, student ID, and their GPA.

```
  def thirdPart(): Unit = {
  val conf = new SparkConf().setAppName("AppName").setMaster("local")
  val sc = new SparkContext(conf)
  val studentRdd = sc.textFile(studentPath).map(_.split(", ", 3))

  studentRdd
    .map { array =>
      // Example 1: B CS201
      // Example 2: A CSC369, B CSC366
      // Handles mapping all courses into numeric values
      val courses = array(2).split(", ")
      val grades = courses
        .map(s => {
          val letter = s.split(" ")(0).trim
          getGrade(letter)
        })

      // Handles GPA calculation
      val gradeSum = grades.aggregate(0.0)(
        (acc, grade) => acc + grade, // Add the value of the grades
        (acc1, acc2) => acc1 + acc2 // Accumulators
      )
      val gpa = gradeSum / courses.length

      // Name, ID, gpa
      (array(0), array(1), gpa)
    }
    .foreach { case (name, id, gpa) =>
      println(s"$name, $id, $gpa")
    }
 }

 def getGrade(grade: String): Double = grade match {
  case "A" => 4.0
  case "B" => 3.0
  case "C" => 2.0
  case "D" => 1.0
  case _ => 0.0
 }
```