Assignment 2

Input File
John Back, 23, B, CSC366
Bob Wilson, 11, B, CS201
John Back, 23, A, CSC369

Format - Input
Student name, Student ID, grade, Course

Format - Output
Student name, Student id, list of classes

Output File
Bob Wilson, 11, (B, CS201)
John Back, 23, (A, CSC369), (B, CSC366) // Sorted by name then grade

a) What is the natural and composite key?
Natural key: Student name, Student id
Composite key: Student name, Student id, Course

b) Composite key Class, Create a StudentCourse pair

```
Public class StudentCoursePair implements WritableComparable <StudentCoursePair> {
    Text studentName = new Text();
    Text course      = new Text();
    // constructor...
    @Override
    Public int compareTo(StudentCoursePair pair) {
    if(this.StudentName.CompareTo(Pair.studentName) == 0)
        return this.course.compareTo(Pair.course);
    return this.StudentName.CompareTo(pair.StudentName);
```

c) Show the mapper class

```
public class StudentMapper extends Mapper <LongWritable, Text, StudentCoursePair, Text> {
  public void map(LongWritable key, Text value, Context context) throws IOException {
    String line = value.toString().trim();
    String[] tokens = line.split(",");
    String nameId = tokens[0] + ", " + tokens[1]; // name and ID
    String course = tokens[2] + ", " + tokens[3];
    Context.Write( new StudentCoursePair(nameId, course), new Text(course));
  }
}
```

Show the partitioner class

```
Public Class StudentPartitioner extends Partitioner (StudentCoursePair, Text) {
@override
 Public int getPartition(StudentCoursePair pair, Text Course, int numPartitions){
    return Math.abs(pair.StudentName.hashCode()% numPartitions);
}
```

Show the group Comparator class

```
Public class StudentGroupingComparator extends WritableComparator {
     Public StudentGroupingComparator() { Super(StudentCoursePair.class, true); }
```

Show the reducer class

```
Public class StudentReducer extends Reducer (StudentCoursePair, Text, Text, Text) {
@override
Protected Void reduce (StudentCoursePair Key , Iterable <Text> Values, Context) {
    String result = "";
    for (Text Value: Values)
        result += (Value.toString() + ",");
    result = result.Substring(0, result.length()-1);
    Context.write( Key.StudentName , new Text(result));
}
```