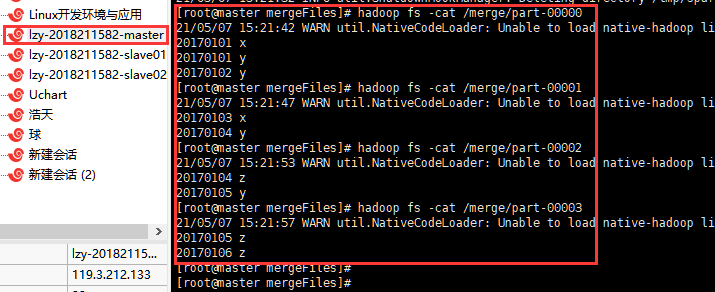
课程实验四：Spark SQL、Spark Streaming

实验时间：2021年05月07日

学生姓名：李志毅

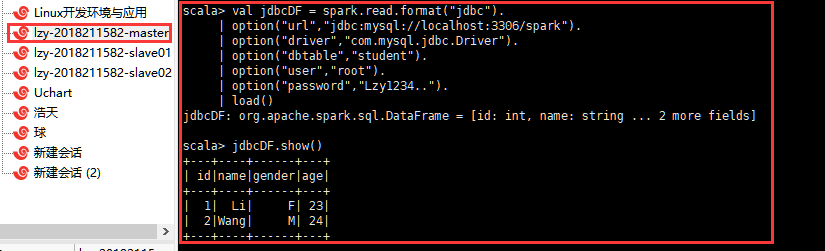
学生班号、学号：2018211314班 2018211582

**一、实验结果截图**

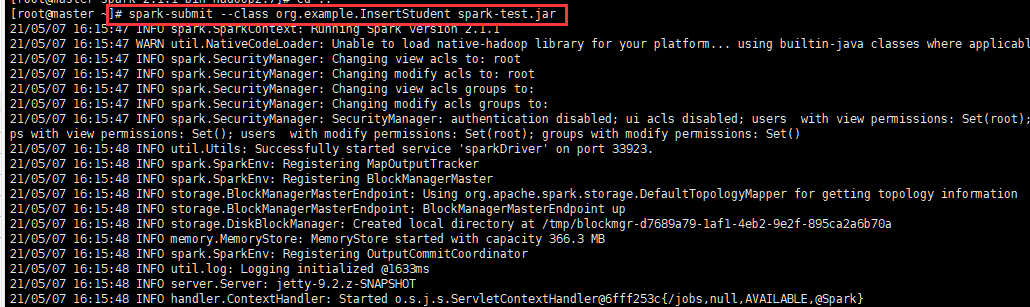
RDD编程实验结果：

**图一：文件合并去重结果**

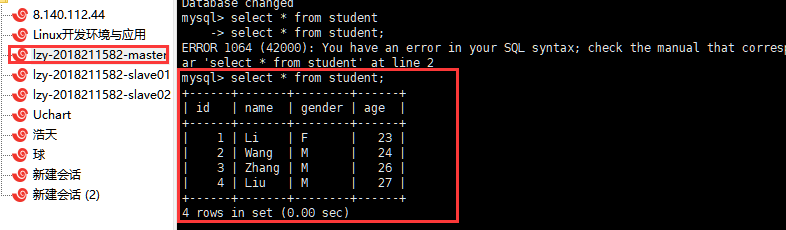
JDBC连接数据库实验结果：



**图二：通过jdbc连接MySQL数据库**

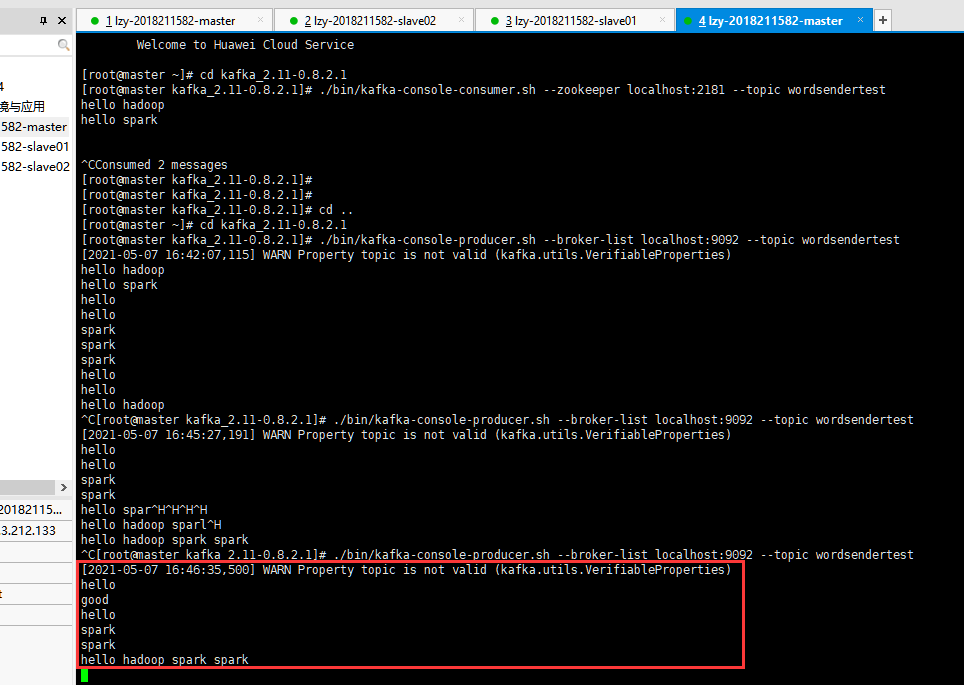


**图三：执行编写好的jar包**

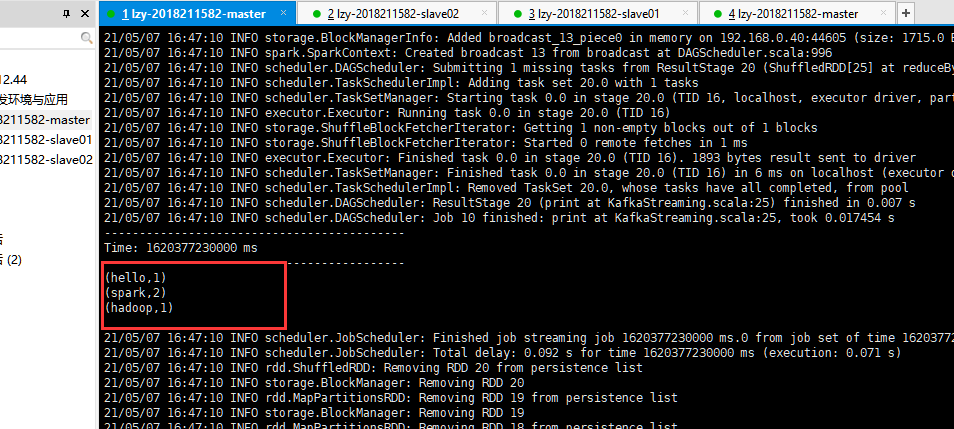


**图四：jar包执行完毕后student表内容**

Spark Streaming实验结果：



**图五：生产者终端展示**



**图六：消费者统计词频结果**

**二、简要描述实验做了哪些工作？**

实验一：编写Scala程序完成了对于两个文本文件数据的合并与去重工作。

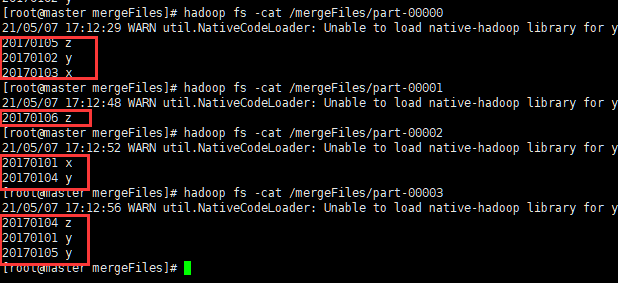
实验二：首先在服务器上安装并配置了MySQL，之后通过编写程序完成了在Spark中通过JDBC连接MySQL数据库，编写Scala程序向MySQL中写入了部分数据。

实验三：首先安装并启动了Kafka程序，之后使用Scala编写了消费者方面Spark Streaming的程序(统计词频)，通过生产者输入词，消费者处理并打印结果，测试了Spark Streaming的工作原理和过程。

**三.实验过程中遇到的问题和解决办法？**

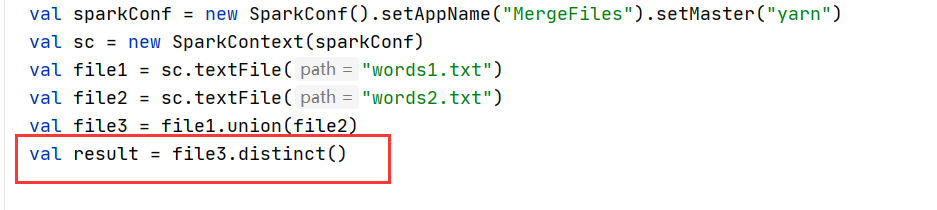
**1.去重后排序问题**

**问题描述：**在第一个实验中，去重之后的结果并没有根据第一列的顺序排序，虽然结果正确，但输出是杂乱无章的

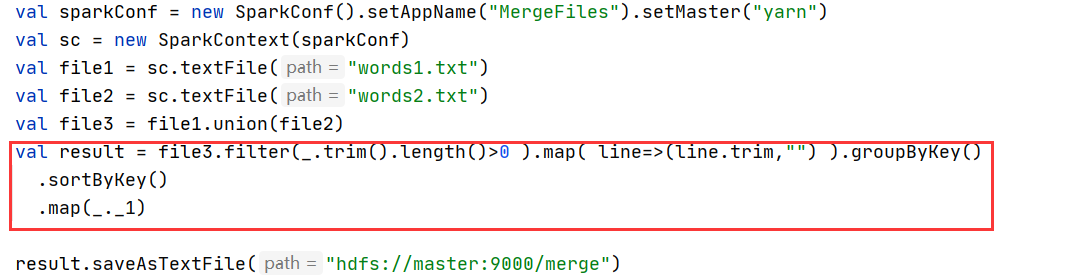


**图-1 未排序结果**

**错误分析：**使用distinct()方法后并没有再次进行排序操作，修改成另一方法，可直接根据第一列进行排序



**图-2 直接使用distinct()方法后不做任何操作**

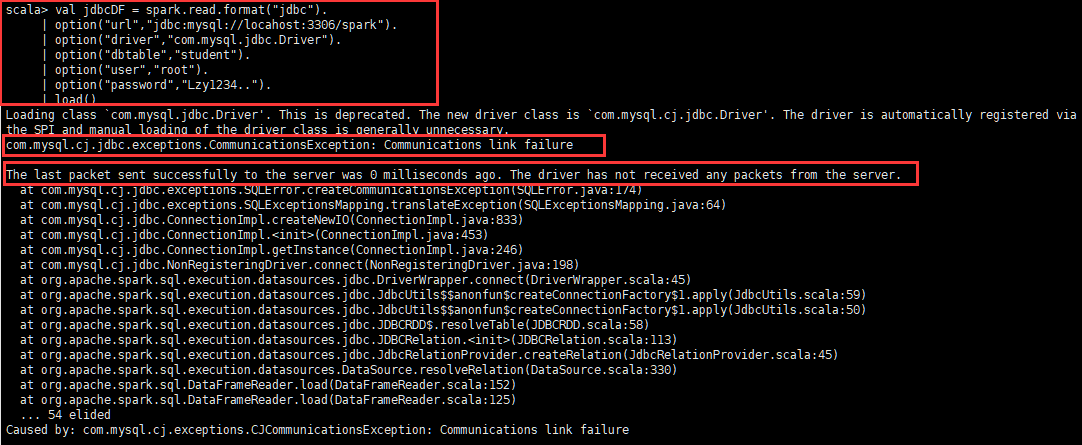


**图-3 修改后程序**

**问题思考：**这个问题暴露出我在做实验时对于每一步操作的检查不够细致，虽然结果正确，但是排序操作耗费了我大量时间，究其原因是对于Scala代码语法的不熟悉，对于Scala程序编写的生疏，这也警示自己做实验前一定要做好充足的准备，在实验过程中也要步步为营，稳扎稳打的进行实验。

**2.Spark连接Mysql数据库出错**

**问题描述：**按照实验指导书启动spark shell后并输入完命令后，报错

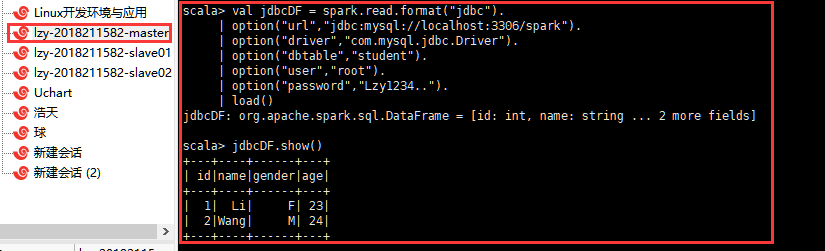


**图-4 连接出错**

**错误分析：**在查看群里的同样问题的解决方案后，发现是

option(“url”,”jdbc:mysql://locahost:3306/spark”)

打错了，localhost少了一个l，修改后连接成功



**图-5 修改后成功**

**问题思考：**实验步骤不够细致，只有需要问题了才反应过来，这种粗心的错误不应该出现，之后做实验时一定要注意。

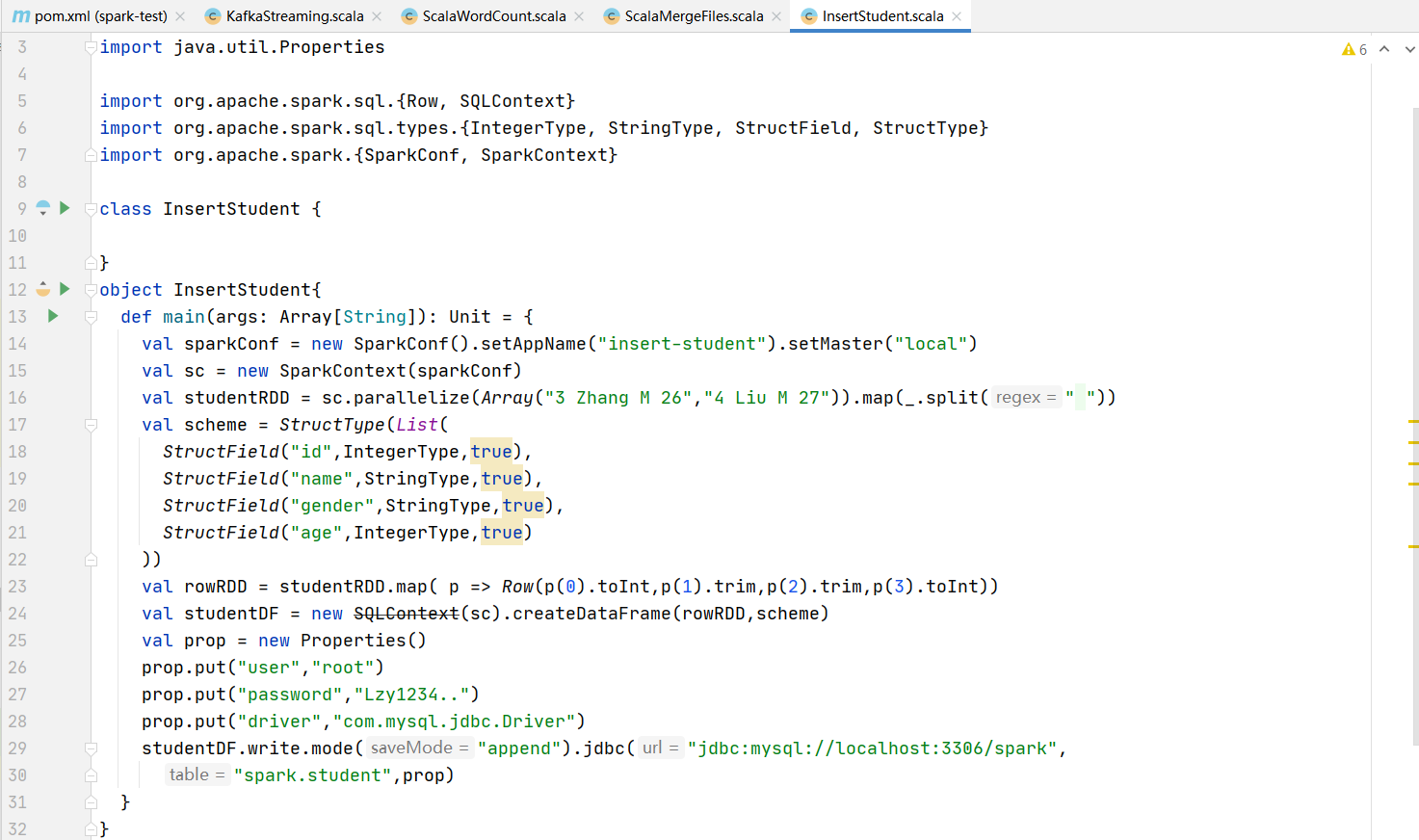
**四.实验代码**

1. RDD编程实验代码



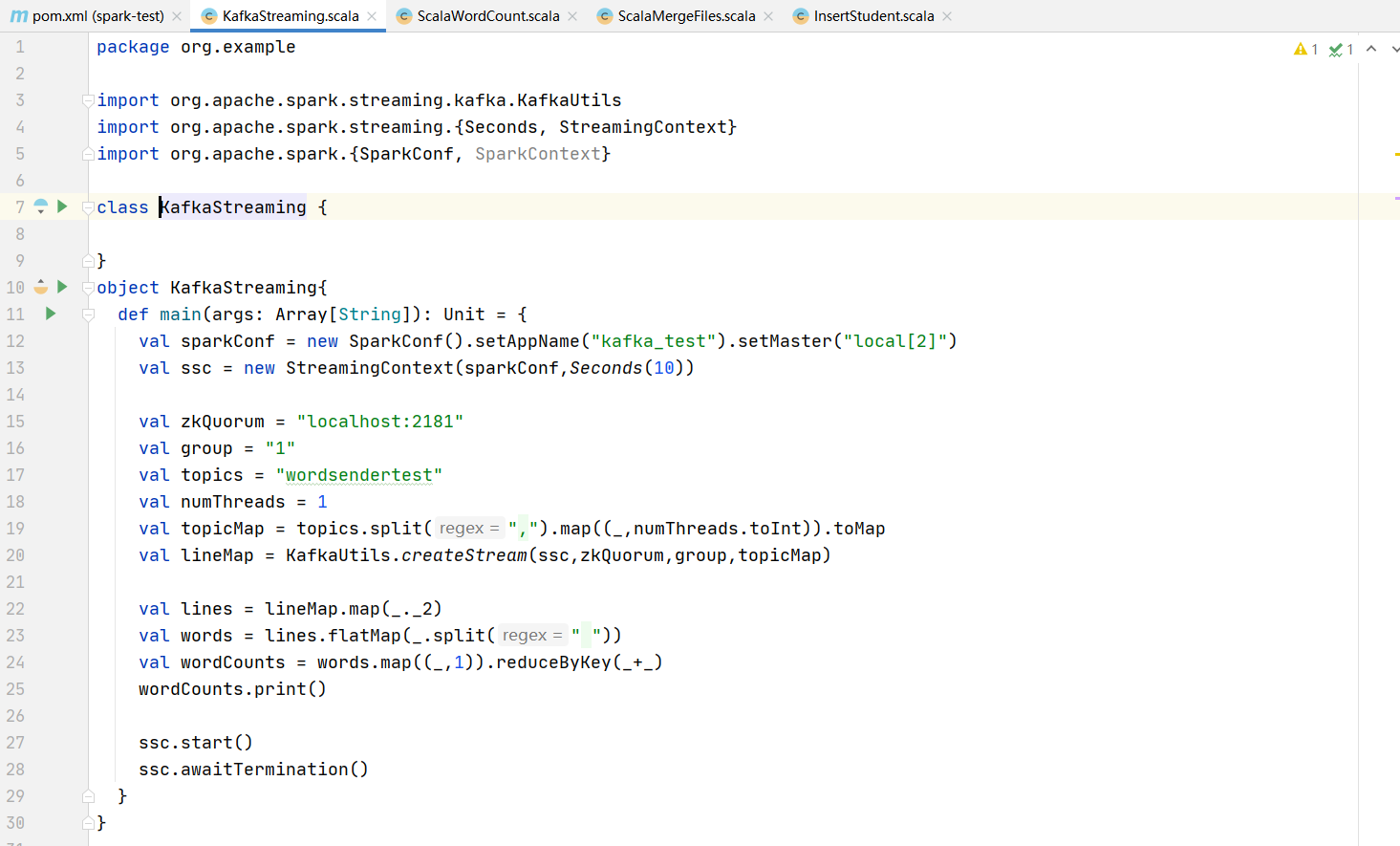
|  |
| --- |
| **package** org.example    **import** org.apache.spark.{ SparkConf, SparkContext}    **class** ScalaMergeFiles {    }  **object** ScalaMergeFiles{  **def** main(args: Array[String]): Unit = {  **val** sparkConf = **new** SparkConf().setAppName("MergeFiles").setMaster("yarn")  **val** sc = **new** SparkContext(sparkConf)  **val** file1 = sc.textFile("words1.txt")  **val** file2 = sc.textFile("words2.txt")  **val** file3 = file1.union(file2)  **val** result = file3.filter(\_.trim().length()>0 ).map( line=>(line.trim,"") ).groupByKey()  .sortByKey()  .map(\_.\_1)    result.saveAsTextFile("hdfs://master:9000/merge")  sc.stop()  }    } |

2.JDBC连接数据库实验代码



|  |
| --- |
| **package** org.example    **import** java.util.Properties    **import** org.apache.spark.sql.{Row, SQLContext}  **import** org.apache.spark.sql.types.{IntegerType, StringType, StructField, StructType}  **import** org.apache.spark.{SparkConf, SparkContext}    **class** InsertStudent {    }  **object** InsertStudent{  **def** main(args: Array[String]): Unit = {  **val** sparkConf = **new** SparkConf().setAppName("insert-student").setMaster("local")  **val** sc = **new** SparkContext(sparkConf)  **val** studentRDD = sc.parallelize(Array("3 Zhang M 26","4 Liu M 27")).map(\_.split(" "))  **val** scheme = StructType(List(  StructField("id",IntegerType,**true**),  StructField("name",StringType,**true**),  StructField("gender",StringType,**true**),  StructField("age",IntegerType,**true**)  ))  **val** rowRDD = studentRDD.map( p => Row(p(0).toInt,p(1).trim,p(2).trim,p(3).toInt))  **val** studentDF = **new** SQLContext(sc).createDataFrame(rowRDD,scheme)  **val** prop = **new** Properties()  prop.put("user","root")  prop.put("password","Lzy1234..")  prop.put("driver","com.mysql.jdbc.Driver")  studentDF.write.mode("append").jdbc("jdbc:mysql://localhost:3306/spark",  "spark.student",prop)  }  } |

3.Spark Streaming实验代码



|  |
| --- |
| **package** org.example    **import** org.apache.spark.streaming.kafka.KafkaUtils  **import** org.apache.spark.streaming.{Seconds, StreamingContext}  **import** org.apache.spark.{SparkConf, SparkContext}    **class** KafkaStreaming {    }  **object** KafkaStreaming{  **def** main(args: Array[String]): Unit = {  **val** sparkConf = **new** SparkConf().setAppName("kafka\_test").setMaster("local[2]")  **val** ssc = **new** StreamingContext(sparkConf,Seconds(10))    **val** zkQuorum = "localhost:2181"  **val** group = "1"  **val** topics = "wordsendertest"  **val** numThreads = 1  **val** topicMap = topics.split(",").map((\_,numThreads.toInt)).toMap  **val** lineMap = KafkaUtils.createStream(ssc,zkQuorum,group,topicMap)    **val** lines = lineMap.map(\_.\_2)  **val** words = lines.flatMap(\_.split(" "))  **val** wordCounts = words.map((\_,1)).reduceByKey(\_+\_)  wordCounts.print()    ssc.start()  ssc.awaitTermination()  }  } |