

## Assignment 26

### Task 1

Read a stream of Strings, fetch the words which can be converted to numbers. Filter out the rows, where the sum of numbers in that line is odd.

Provide the sum of all the remaining numbers in that batch.

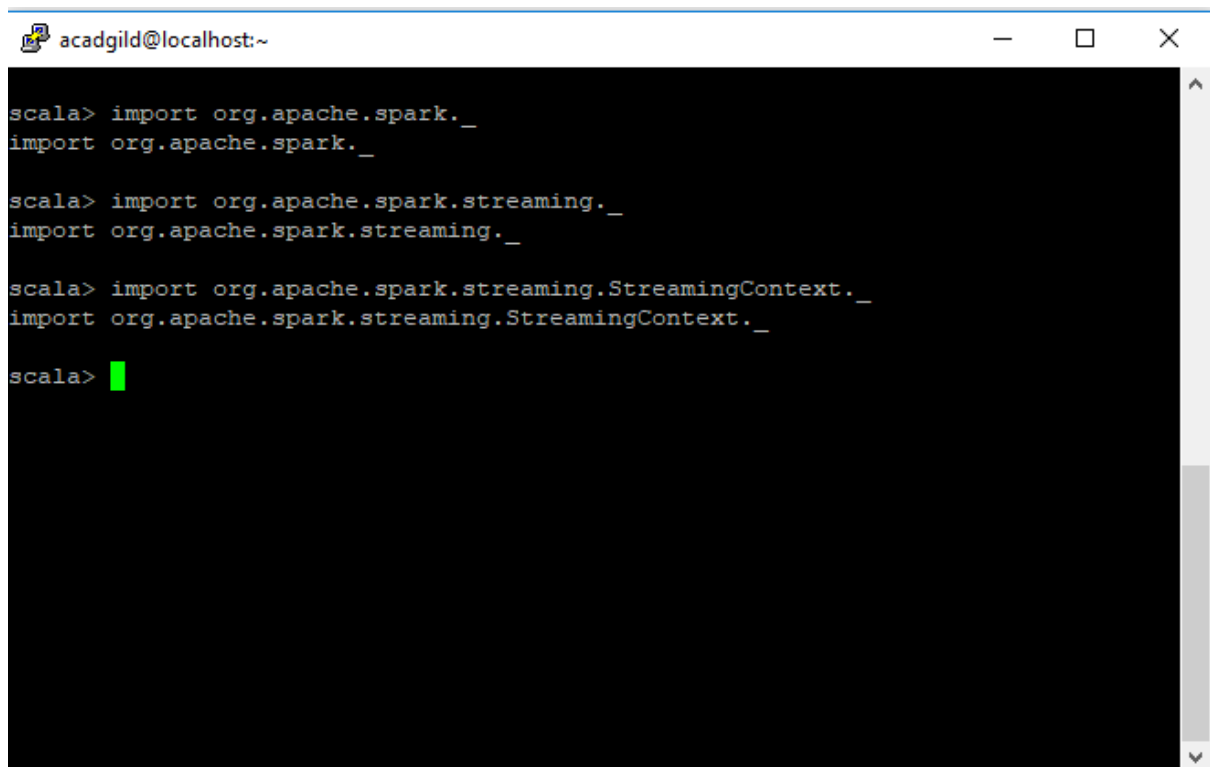
Step 1:

Declare all the packages

`import org.apache.spark._`

`import org.apache.spark.streaming._`

`import org.apache.spark.streaming.StreamingContext._`

A screenshot of a Scala REPL window titled 'acadgild@localhost:~'. The window shows the following code being entered:

```
scala> import org.apache.spark._
import org.apache.spark._

scala> import org.apache.spark.streaming._
import org.apache.spark.streaming._

scala> import org.apache.spark.streaming.StreamingContext._
import org.apache.spark.streaming.StreamingContext._

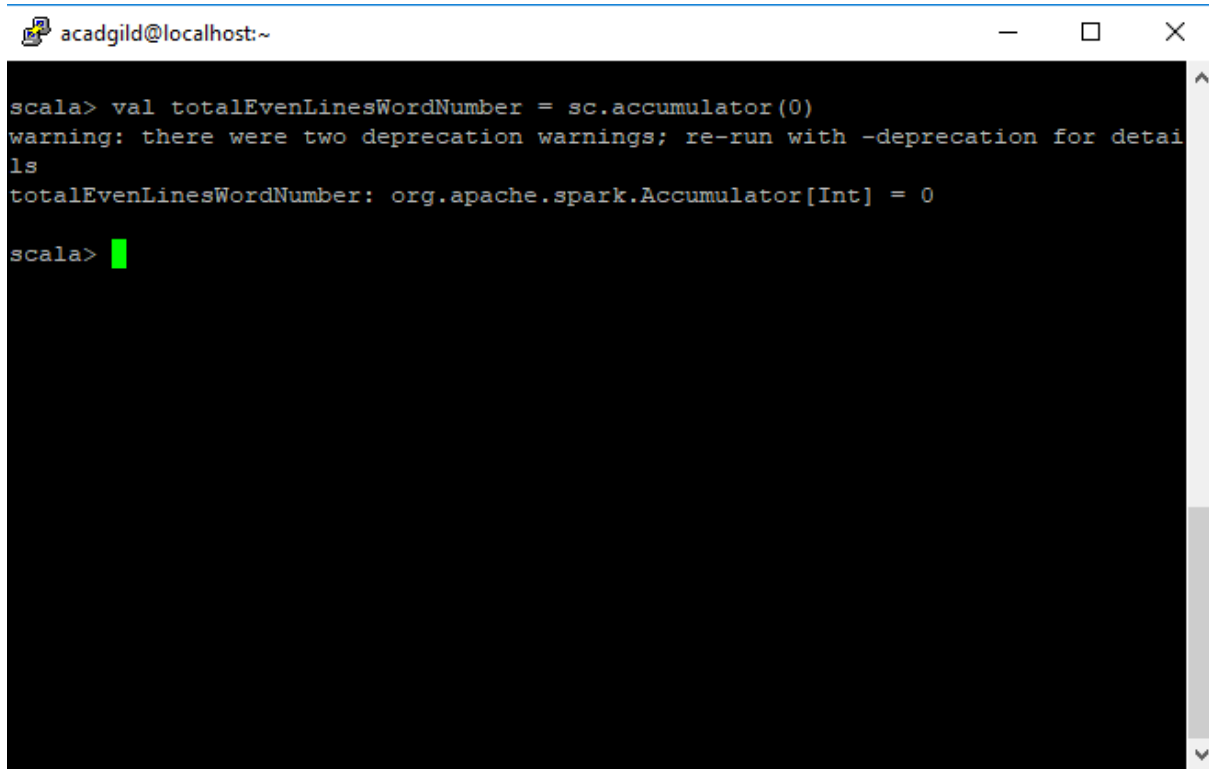
scala> 
```

The cursor is at the end of the last line, ready for the next command.

Step 2 :

Declare accumulator `totalEvenLinesWordNumber` which will keep track of sum of number of word numbers in lines so far

`val totalEvenLinesWordNumber = sc.accumulator(0)`

A terminal window with a black background and white text. The window title bar shows 'acadgild@localhost:~' and standard window controls. The text inside the terminal shows a Scala REPL session where an accumulator is created and its value is displayed. The prompt 'scala>' is followed by a green cursor.

```
acadgild@localhost:~  
scala> val totalEvenLinesWordNumber = sc.accumulator(0)  
warning: there were two deprecation warnings; re-run with -deprecation for details  
totalEvenLinesWordNumber: org.apache.spark.Accumulator[Int] = 0  
scala> █
```

### Step 3:

Define a `wordNumberMap` map for converting word to number

```
val wordNumberMap = Map("Hi" -> 1, "my" -> 2, "name" -> 3, "is" -> 4,  
"Monisha" ->5, "Acadgild" ->9)
```

```
val wordNumberMapBroadcast = sc.broadcast(wordNumberMap)
```

```
acadgild@localhost:~  
scala> val totalEvenLinesWordNumber = sc.accumulator(0)  
warning: there were two deprecation warnings; re-run with -deprecation for details  
totalEvenLinesWordNumber: org.apache.spark.Accumulator[Int] = 0  
  
scala> val wordNumberMap = Map("Hi" -> 1, "my" -> 2, "name" -> 3, "is" -> 4, "Hello" -> 5, "Mohammed" -> 6, "Fatha" -> 7, "ulla" -> 8, "Acadgild" -> 9)  
wordNumberMap: scala.collection.immutable.Map[String,Int] = Map(name -> 3, is -> 4, ulla -> 8, my -> 2, Hello -> 5, Mohammed -> 6, Hi -> 1, Acadgild -> 9, Fatha -> 7)  
  
scala> val wordNumberMapBroadcast = sc.broadcast(wordNumberMap)  
wordNumberMapBroadcast: org.apache.spark.broadcast.Broadcast[scala.collection.immutable.Map[String,Int]] = Broadcast(0)  
  
scala> █
```

#### Step 4:

Define a function to return sum of word converted to number in a line

```
def lineWordNumberTotal(line:String):Int = {  
    var sum:Int = 0  
    var words = line.split(" ")  
    for (word <- words) sum +=  
wordNumberMapBroadcast.value.get(word).getOrElse(0)  
    sum  
}
```

```
acadgild@localhost:~  
warning: there were two deprecation warnings; re-run with -deprecation for details  
totalEvenLinesWordNumber: org.apache.spark.Accumulator[Int] = 0  
  
scala> val wordNumberMap = Map("Hi" -> 1, "my" -> 2, "name" -> 3, "is" -> 4, "Hello" -> 5, "Mohammed" -> 6, "Fatha" -> 7, "ulla" -> 8, "Acadgild" -> 9)  
wordNumberMap: scala.collection.immutable.Map[String,Int] = Map(name -> 3, is -> 4, ulla -> 8, my -> 2, Hello -> 5, Mohammed -> 6, Hi -> 1, Acadgild -> 9, Fatha -> 7)  
  
scala> val wordNumberMapBroadcast = sc.broadcast(wordNumberMap)  
wordNumberMapBroadcast: org.apache.spark.broadcast.Broadcast[scala.collection.immutable.Map[String,Int]] = Broadcast(0)  
  
scala> def lineWordNumberTotal(line:String):Int = {  
  |   var sum:Int = 0  
  |   var words = line.split(" ")  
  |   for (word <- words) sum += wordNumberMapBroadcast.value.get(word).getOrElse(0)  
  |   sum  
  | }  
lineWordNumberTotal: (line: String)Int  
  
scala> █
```

## Step 5:

Start text streaming on localhost with port number 9999 and interval 15 seconds and return the stream

```
val ssc = new StreamingContext(sc, Seconds(15))
```

```
val stream = ssc.socketTextStream("localhost", 9999)
```

```
acadgild@localhost:~  
-> 7)  
  
scala> val wordNumberMapBroadcast = sc.broadcast(wordNumberMap)  
wordNumberMapBroadcast: org.apache.spark.broadcast.Broadcast[scala.collection.immutable.Map[String,Int]] = Broadcast(0)  
  
scala> def lineWordNumberTotal(line:String):Int = {  
  |   var sum:Int = 0  
  |   var words = line.split(" ")  
  |   for (word <- words) sum += wordNumberMapBroadcast.value.get(word).getOrElse(0)  
  |   sum  
  | }  
lineWordNumberTotal: (line: String)Int  
  
scala> val ssc = new StreamingContext(sc, Seconds(15))  
ssc: org.apache.spark.streaming.StreamingContext = org.apache.spark.streaming.StreamingContext@549debdb  
  
scala> val stream = ssc.socketTextStream("localhost", 9999)  
stream: org.apache.spark.streaming.dstream.ReceiverInputDStream[String] = org.apache.spark.streaming.dstream.SocketInputDStream@df34b01  
  
scala>
```

## Step 6:

Process each RDD in stream. First convert the RDD to string.

```
stream.foreachRDD(line => {  
  val lineStr = line.collect().toList.mkString("")  
  if (lineStr != "") {  
    var numTotal = lineWordNumberTotal(lineStr)  
    if (numTotal % 2 == 1) println(lineStr)  
    else {  
      totalEvenLinesWordNumber += numTotal  
      println("Sum of lines with even word number so far =" +  
totalEvenLinesWordNumber.value.toInt)  
    }  
  }  
})
```

```
acadgild@localhost:~  
lineWordNumberTotal: (line: String)Int  
  
scala> val ssc = new StreamingContext(sc, Seconds(15))  
ssc: org.apache.spark.streaming.StreamingContext = org.apache.spark.streaming.StreamingContext@549debdb  
  
scala> val stream = ssc.socketTextStream("localhost", 9999)  
stream: org.apache.spark.streaming.dstream.ReceiverInputDStream[String] = org.apache.spark.streaming.dstream.SocketInputDStream@df34b01  
  
scala> stream.foreachRDD(line => {  
  |   val lineStr = line.collect().toList.mkString("")  
  |   if (lineStr != "") {  
  |     var numTotal = lineWordNumberTotal(lineStr)  
  |     if (numTotal % 2 == 1) println(lineStr)  
  |     else {  
  |       totalEvenLinesWordNumber += numTotal  
  |       println("Sum of lines with even word number so far =" + totalEvenLinesWordNumber.value.toInt)  
  |     }  
  |   }  
  | })  
  
scala> █
```

**Step 7:**

**Start the streams**

**ssc.start()**

**ssc.awaitTermination()**

```
Scala REPL window titled "Spark Streaming Context" showing the following code and output:

scala> val stream = ssc.socketTextStream("localhost", 9999)
stream: org.apache.spark.streaming.dstream.ReceiverInputDStream[String] = org.ap
ache.spark.streaming.dstream.SocketInputDStream@cce672c

scala>

scala> stream.foreachRDD(line => {
  |   val lineStr = line.collect().toList.mkString("")
  |   if (lineStr != "") {
  |     var numTotal = lineWordNumberTotal(lineStr)
  |     if (numTotal % 2 == 1) println(lineStr)
  |     else {
  |       totalEvenLinesWordNumber += numTotal
  |       println("Sum of lines with even word number so far =" + tota
lEvenLinesWordNumber.value.toInt)
  |     }
  |   }
  | })

scala> ssc.start()

scala> ssc.awaitTermination()
```

**Step 8:**

**Start netcat from a terminal**

**nc -lk 9999**

**Step 9: Display the output**

**Hi = 1 -> It is a odd number so line is display**

**Hello = 5 -> It is a odd number so line is display**

**Hello Acadgild = 5 + 9 = 14 -> It is even number so sum of lines with even word number so far will be displayed**

```

18/04/30 11:38:59 WARN storage.RandomBlockReplicationPolicy: Expecting 1 replicas with only 0 peer/s.
18/04/30 11:38:59 WARN storage.BlockManager: Block input-0-1525068538800 replicated to only 0 peer(s) instead of 1 peers
Hi
18/04/30 11:39:13 WARN storage.RandomBlockReplicationPolicy: Expecting 1 replicas with only 0 peer/s.
18/04/30 11:39:13 WARN storage.BlockManager: Block input-0-1525068553400 replicated to only 0 peer(s) instead of 1 peers
Hi Mohammed
18/04/30 11:39:34 WARN storage.RandomBlockReplicationPolicy: Expecting 1 replicas with only 0 peer/s.
18/04/30 11:39:34 WARN storage.BlockManager: Block input-0-1525068574400 replicated to only 0 peer(s) instead of 1 peers
Sum of lines with even word number so far =40
18/04/30 11:39:54 WARN storage.RandomBlockReplicationPolicy: Expecting 1 replicas with only 0 peer/s.
18/04/30 11:39:54 WARN storage.BlockManager: Block input-0-1525068594600 replicated to only 0 peer(s) instead of 1 peers
Hello
18/04/30 11:40:06 WARN storage.RandomBlockReplicationPolicy: Expecting 1 replicas with only 0 peer/s.
18/04/30 11:40:06 WARN storage.BlockManager: Block input-0-1525068605800 replicated to only 0 peer(s) instead of 1 peers
Sum of lines with even word number so far =54

```

## Task 2

### Read two streams

1. List of strings input by user
2. Real-time set of offensive words

Find the word count of the offensive words inputted by the user as per the real-time set of offensive words

#### Step 1:

```

import org.apache.spark.{SparkConf, SparkContext}
import org.apache.spark.rdd.RDD
import org.apache.spark.streaming.{Seconds, StreamingContext, Time}
import org.apache.spark.sql.SparkSession
import org.apache.log4j.{Level, Logger}

```

```

object SqlNetworkWordCount {

```

```

  def main(args: Array[String]): Unit = {
    println("hey Spark SQL Streaming")
  }

```



```

val conf = new
SparkConf().setMaster("local[2]").setAppName("SparkSteamingExample")

val sc = new SparkContext(conf)

val rootLogger = Logger.getRootLogger()
rootLogger.setLevel(Level.ERROR)

println("hey Spark Streaming ---> 1")

//val sparkConf = new SparkConf().setAppName("NetworkWordCount")
println("hey Spark Streaming ---> 2")

val ssc = new StreamingContext(sc, Seconds(10))
val lines = ssc.socketTextStream("localhost", 9999)
println("hey Spark Streaming ---> 3")
val words = lines.flatMap(_.split(" "))

// Convert RDDs of the words DStream to DataFrame and run SQL query
words.foreachRDD { (rdd: RDD[String], time: Time) =>
    val spark = SparkSessionSingleton.getInstance(rdd.sparkContext.getConf)
    import spark.implicits._

    // Convert RDD[String] to RDD[case class] to DataFrame
    val wordsDataFrame = rdd.map(w => Record(w)).toDF()

    // Creates a temporary view using the DataFrame
    wordsDataFrame.createOrReplaceTempView("words")

    // Do word count on table using SQL and print it
    val wordCountsDataFrame =
        spark.sql("select word, count(*) as total from words group by word")
    println(s"===== $time =====")
}

```

```
    wordCountsDataFrame.show()
  }
  ssc.start()
  ssc.awaitTermination()
}

/** Case class for converting RDD to DataFrame */
case class Record(word: String)

/** Lazily instantiated singleton instance of SparkSession */
object SparkSessionSingleton {
  @transient private var instance: SparkSession = _

  def getInstance(sparkConf: SparkConf): SparkSession = {
    if (instance == null) {
      instance = SparkSession
        .builder
        .config(sparkConf)
        .getOrCreate()
    }
    instance
  }
}
```

```
SqlNetworkWordCount.scala
1
2+ import org.apache.spark.{SparkConf, SparkContext}
7
8= object SqlNetworkWordCount {
9
10= def main(args: Array[String]): Unit = {
11     println("hey Spark SQL Streaming");
12
13     val conf = new SparkConf().setMaster("local[2]").setApp
14     val sc = new SparkContext(conf);
15     val rootLogger = Logger.getRootLogger();
16     rootLogger.setLevel(Level.ERROR);
17
18
19     println("hey Spark Streaming ---> 1");
20     //val sparkConf = new SparkConf().setAppName("NetworkW
21     println("hey Spark Streaming ---> 2");
22     val ssc = new StreamingContext(sc, Seconds(10));
23     val lines = ssc.socketTextStream("localhost", 9999);
24     println("hey Spark Streaming ---> 3");
```

```
SqlNetworkWordCount.scala
25
26     val words = lines.flatMap(_.split(" "));
27
28     // Convert RDDs of the words DStream to DataFrame and
29     words.foreachRDD { (rdd: RDD[String], time: Time) =>
30         val spark = SparkSessionSingleton.getInstance(rdd.sp
31         import spark.implicits._;
32
33         // Convert RDD[String] to RDD[case class] to DataFrame
34         val wordsDataFrame = rdd.map(w => Record(w)).toDF();
35
36         // Creates a temporary view using the DataFrame
37         wordsDataFrame.createOrReplaceTempView("words");
38
39         // Do word count on table using SQL and print it
40         val wordCountsDataFrame =
41         | spark.sql("select word, count(*) as total from words")
42         println(s"===== $time =====")
43         wordCountsDataFrame.show()
44     }
45
```

```
SqlNetworkWordCount.scala  ✖
46
47     ssc.start()
48     ssc.awaitTermination()
49
50 }
51
52
53 /** Case class for converting RDD to DataFrame */
54 case class Record(word: String)
55
56 /** Lazily instantiated singleton instance of SparkSession */
57 object SparkSessionSingleton {
58
59     @transient private var instance: SparkSession = _
60
61     def getInstance(sparkConf: SparkConf): SparkSession = {
62         if (instance == null) {
63             instance = SparkSession
64                 .builder
65                 .config(sparkConf)
```

```
SqlNetworkWordCount.scala  ✖
55
56 /** Lazily instantiated singleton instance of SparkSession */
57 object SparkSessionSingleton {
58
59     @transient private var instance: SparkSession = _
60
61     def getInstance(sparkConf: SparkConf): SparkSession = {
62         if (instance == null) {
63             instance = SparkSession
64                 .builder
65                 .config(sparkConf)
66                 .getOrCreate()
67         }
68         instance
69     }
70 }
71
72 }
73
74
```

**Step 2: Start netcat from a terminal**

**nc -lk 9999**

```
acadgild@localhost:~  
[acadgild@localhost ~]$ nc -lk 9999  
word  
Hey Acadgild  
wordcount wordcount wordcount  
Bigdata Bigdata Bigdata Bigdata
```

### Step 3: Display Results:

**word = 1**

**Hey Acadgild = 1**

**Wordcount = 3**

**Bigdata = 4**

```
Problems Tasks Console  
SqlNetworkWordCount$ [Scala Application] /usr/java/jdk1.8.0_151/bin/java (Ap  
===== 1525072840000 ms =====  
+-----+  
|word|total|  
+-----+  
|word|    1|  
+-----+  
  
===== 1525072850000 ms =====  
+-----+  
|    word|total|  
+-----+  
|Acadgild|    1|  
|    Hey|    1|  
+-----+
```

```
Problems Tasks Console
SqlNetworkWordCount$ [Scala Application] /usr/java/jdk1.8.0_151/bin/java (Apr 30, 2016)

+-----+-----+

===== 1525072890000 ms =====
+-----+-----+
|word|total|
+-----+-----+
+-----+-----+

===== 1525072900000 ms =====
+-----+-----+
|      word|total|
+-----+-----+
|wordcount|      3|
+-----+-----+
```

```
Problems Tasks Console
SqlNetworkWordCount$ [Scala Application] /usr/java/jdk1.8.0_151/bin/java (Apr 30, 2016)

+-----+-----+

===== 1525072940000 ms =====
+-----+-----+
|word|total|
+-----+-----+
+-----+-----+

===== 1525072950000 ms =====
+-----+-----+
|  word|total|
+-----+-----+
|Bigdata|      4|
+-----+-----+
```

