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.

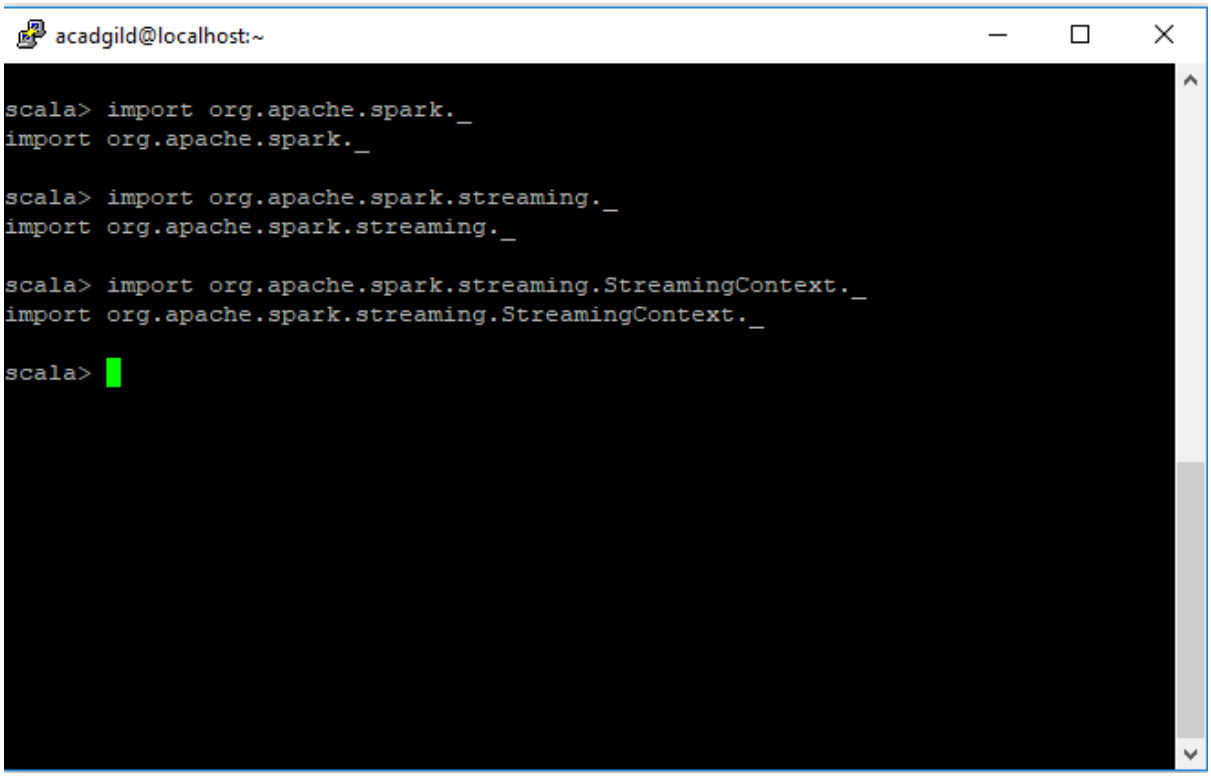
Step 1:

Declare all the packages

import org.apache.spark._

import org.apache.spark.streaming._

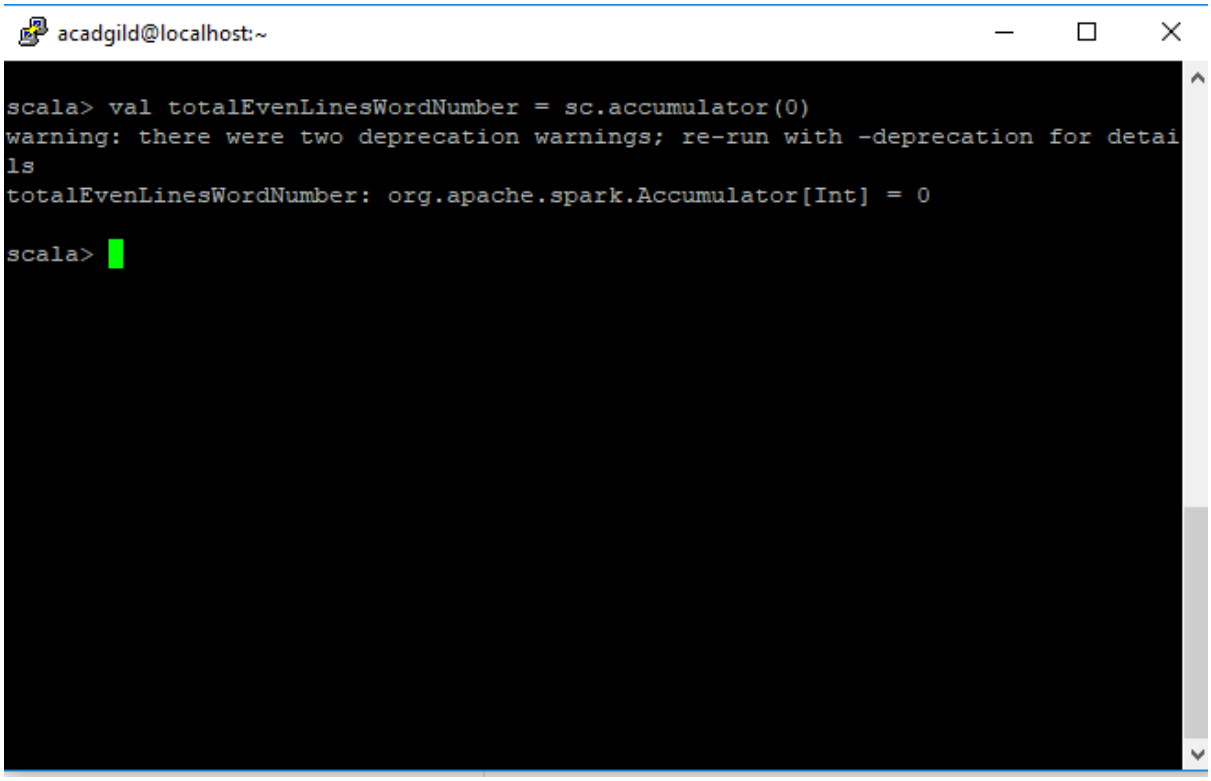
import org.apache.spark.streaming.StreamingContext._



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)



Step 3:

Define a wordNumberMap map for converting word to number

val wordNumberMap = Map("Hi" -> 1, "my" -> 2, "name" -> 3, "is" -> 4, "Hello" ->5, "Mohammed" -> 6, "Fatha" -> 7, "ulla"->8, "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 detai
ls
totalEvenLinesWordNumber: org.apache.spark.Accumulator[Int] = 0

scala> val wordNumberMap = Map("Hi" -> 1, "my" -> 2, "name" -> 3, "is" -> 4, "He
llo" ->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.im
mutable.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 detai
ls
totalEvenLinesWordNumber: org.apache.spark.Accumulator[Int] = 0

scala> val wordNumberMap = Map("Hi" -> 1, "my" -> 2, "name" -> 3, "is" -> 4, "He
llo" ->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.im
mutable.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).get
OrElse(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.im
mutable.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).get
OrElse(0)
     |     sum
     | }
lineWordNumberTotal: (line: String)Int

scala> val ssc = new StreamingContext(sc, Seconds(15))
ssc: org.apache.spark.streaming.StreamingContext = org.apache.spark.streaming.St
reamingContext@549debdb

scala> val stream = ssc.socketTextStream("localhost", 9999)
stream: org.apache.spark.streaming.dstream.ReceiverInputDStream[String] = org.ap
ache.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.St
reamingContext@549debdb

scala> val stream = ssc.socketTextStream("localhost", 9999)
stream: org.apache.spark.streaming.dstream.ReceiverInputDStream[String] = org.ap
ache.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 =" + tota
lEvenLinesWordNumber.value.toInt)
     |          }
     |      }
     | } )

scala>
```

Step 7:

Start the streams

ssc.start()

ssc.awaitTermination()

```
acadgild@localhost:~
reamingContext@19bbb216

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

```
acadgild@localhost:~
[acadgild@localhost ~]$ nc -lk 9999
Hi
Hi Mohammed
Hi Fatha
Hello
Hello Acadgild
```

Step 9: Display the output

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

Hi Mohammed = 1 + 6 = 7 -> It is a odd number so line is display

Hi Fatha = 1 + 7 = 8 -> It is even number so sum of lines with even word number so far will be displayed

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

```scala

import org.apache.spark.{SparkConf, SparkContext}

object SqlNetworkWordCount {

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

    val conf = new SparkConf().setMaster("local[2]").setAp
    val sc = new SparkContext(conf);
   val rootLogger =Logger.getRootLogger();
   rootLogger.setLevel(Level.ERROR);


    println("hey Spark Streaming ---> 1");
    //val sparkConf = new SparkConf().setAppName("NetworkW
    println("hey Spark Streaming ---> 2");
    val ssc = new StreamingContext(sc, Seconds(10));
    val lines = ssc.socketTextStream("localhost", 9999);
    println("hey Spark Streaming ---> 3");
```

SqlNetworkWordCount.scala

```scala
    val words = lines.flatMap(_.split(" "));

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

      // Convert RDD[String] to RDD[case class] to DataFra
      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 wor
      println(s"========= $time =========")
      wordCountsDataFrame.show()
    }
```

SqlNetworkWordCount.scala

```scala
    ssc.start()
    ssc.awaitTermination()


  }

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

  /** Lazily instantiated singleton instance of SparkSessi
  object SparkSessionSingleton {

    @transient private var instance: SparkSession = _

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

SqlNetworkWordCount.scala

```scala

  /** Lazily instantiated singleton instance of SparkSessi
  object SparkSessionSingleton {

    @transient private var instance: SparkSession = _

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

}
```

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 = 1

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 3
+----+-----+

========= 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, 201
+----+-----+

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

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