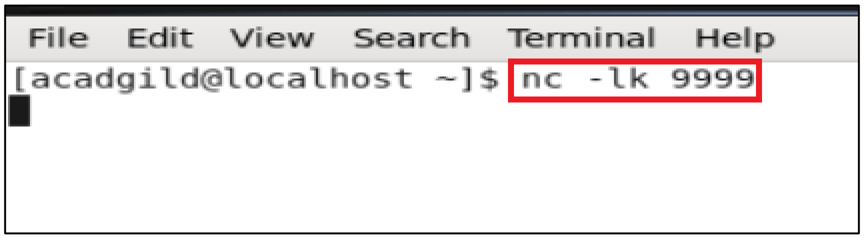
Session 26 – Spark Streaming

Assignment 1

**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.**

For reading the stream, the stream is created using the netcat utility. A TCP socket stream is created by running netcat as a data server.

nc -lk 9999

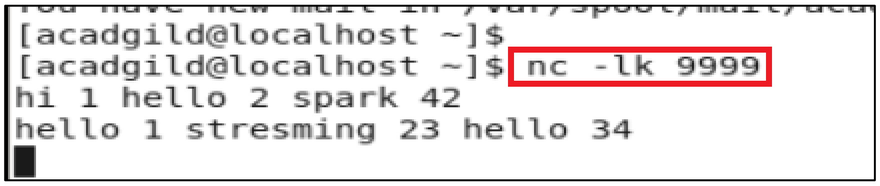


```
[acadgild@localhost ~]$ nc -lk 9999
```

The spark program for read the stream and displaying the required data is below:

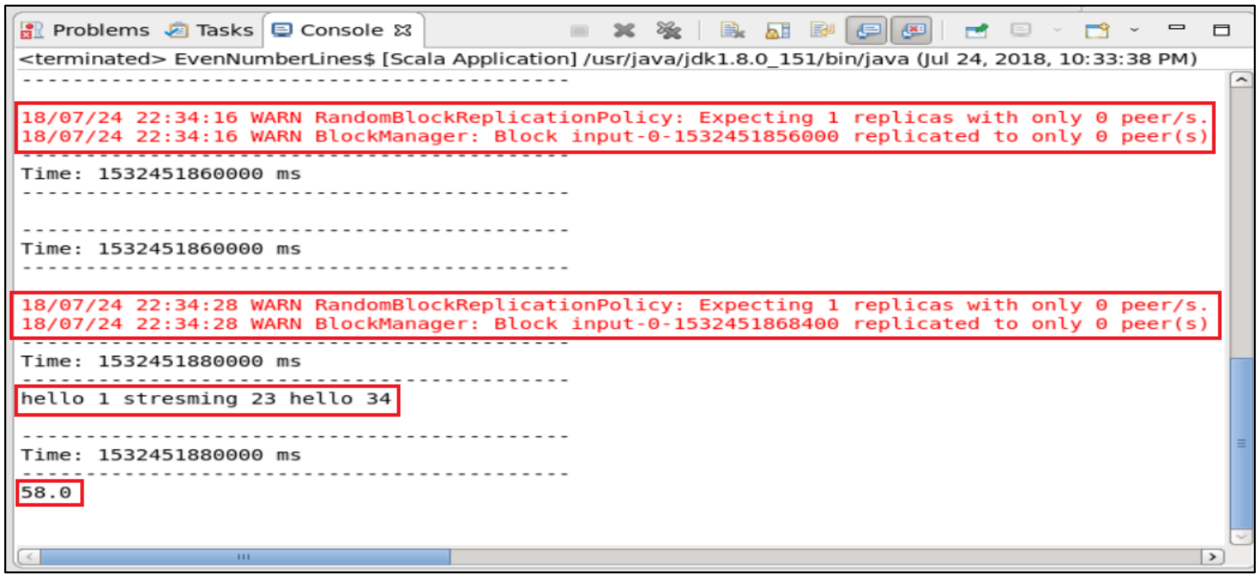
```
package com.acadgild.sparkstreaming.task1

import org.apache.spark.{SparkConf,SparkContext}
import org.apache.spark.streaming.{Seconds, StreamingContext}

object EvenNumberLines {
  def main(args: Array[String]) : Unit = {
    def GetLinesSum(input: String) : Double={

      //spliiting the input line on spaces as delimiter
      val line = input.split(" ")
      var number : Double = 0.0
      for (x <- line)
      {
        try{
          // converting the words to double.
          val value = x.toDouble
          //calculating the sum of the numbers.
          number = number+ value
        }
        catch{
          case ex :  Exception => {}
        }
      }
      return number;
    }
    val conf = new SparkConf().setMaster("local[2]").setAppName("EvenLines")
    val sc = new SparkContext(conf)

    sc.setLogLevel("WARN")
    println("Spark COntext Created")

    // Create a local StreamingContext with working thread and batch interval of 20
seconds.
    val ssc = new StreamingContext(sc,Seconds(20))

    println("Spark Streaming Context created")

   // Create a DStream that will connect to hostname:port,localhost:9999
    val lines =  ssc.socketTextStream("localhost", 9999)

   // filtering only the entries where the sum is not odd, i.e even
    val lines_filter = lines.filter(x => GetLinesSum(x)%2 ==0)

    val lines_sum = lines_filter.map(x => GetLinesSum(x))

   lines_filter.print()

   lines_sum.reduce(_+_).print()

    ssc.start()

    ssc.awaitTermination()
  }

}
```

After executing the program, the stream is initiated by entering data through the port 9999



```
[acadgild@localhost ~]$
[acadgild@localhost ~]$ nc -lk 9999
hi 1 hello 2 spark 42
hello 1 stresming 23 hello 34
```

The program reads the data as soon as every line is entered in the stream. Then the filtering is done and only the records which have the sum as even number is displayed.



```
<terminated> EvenNumberLines$ [Scala Application] /usr/java/jdk1.8.0_151/bin/java (Jul 24, 2018, 10:33:38 PM)
-------------------------------------------
18/07/24 22:34:16 WARN RandomBlockReplicationPolicy: Expecting 1 replicas with only 0 peer/s.
18/07/24 22:34:16 WARN BlockManager: Block input-0-1532451856000 replicated to only 0 peer(s)
-------------------------------------------
Time: 1532451860000 ms
-------------------------------------------

-------------------------------------------
Time: 1532451860000 ms
-------------------------------------------

18/07/24 22:34:28 WARN RandomBlockReplicationPolicy: Expecting 1 replicas with only 0 peer/s.
18/07/24 22:34:28 WARN BlockManager: Block input-0-1532451868400 replicated to only 0 peer(s)
-------------------------------------------
Time: 1532451880000 ms
-------------------------------------------
hello 1 stresming 23 hello 34

-------------------------------------------
Time: 1532451880000 ms
-------------------------------------------
58.0
```

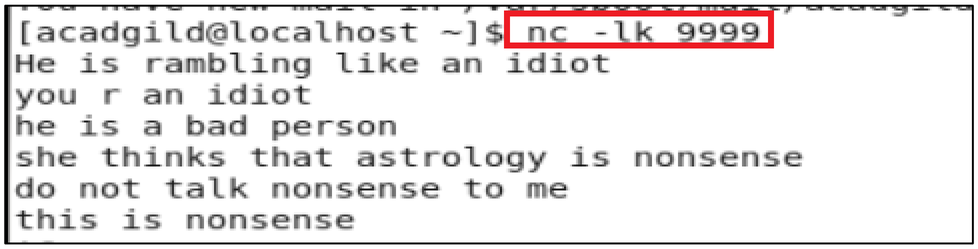
As there are 2 entries in the stream, the stream is read twice.
But nothing is displayed the first time because, the sum is odd and hence gets filtered out.
But for the second entry, the sum is even hence the record as well as the sum is displayed.

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

The spark program for taking in and processing the stream is as follows

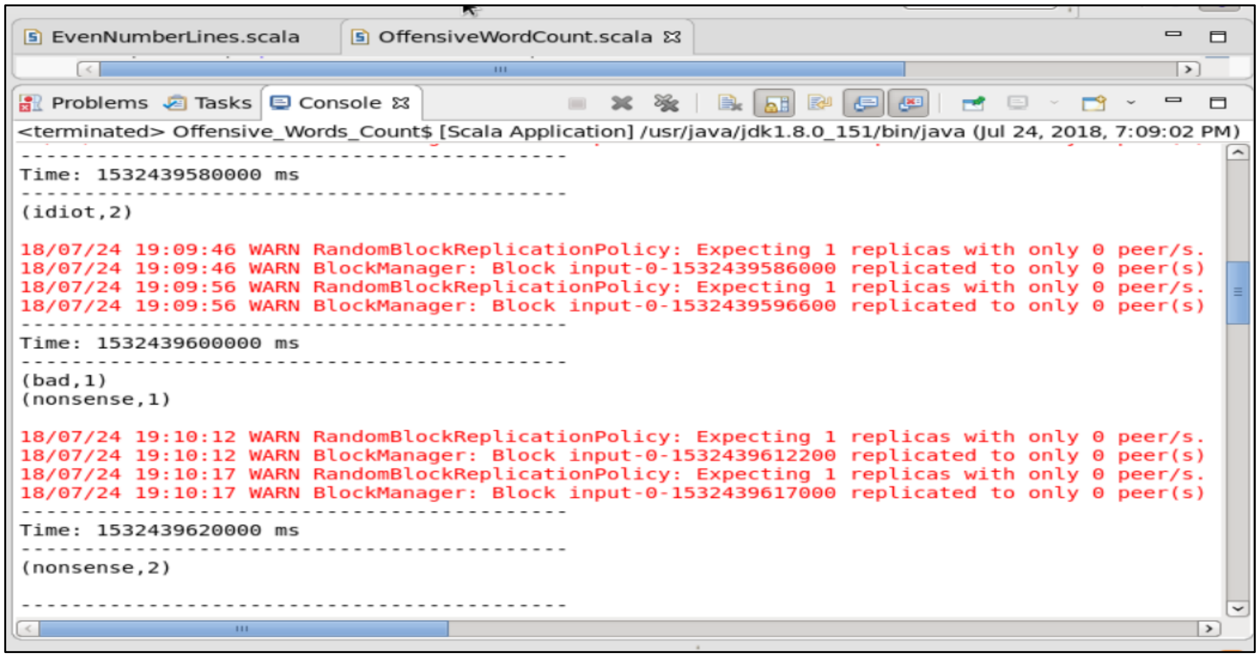
```
package com.acadgild.sparkstreaming.task2

import org.apache.spark.{SparkConf, SparkContext}
import org.apache.spark.streaming.{Seconds, StreamingContext}

object Offensive_Words_Count {

  def main(args: Array[String]): Unit = {
    val conf = new
SparkConf().setMaster("local[2]").setAppName("SparkSteamingExample")
    val sc = new SparkContext(conf)

    sc.setLogLevel("WARN")

    println("Spark Context Created")

    //create a set of offensive words which we use to compare and filter these words from
input string
    val offensive_word_list: Set[String] = Set("idiot", "fool", "bad","nonsense")

    //print the list of these offensive words
    println(s"$offensive_word_list")

    // Create a local StreamingContext with working thread and batch interval of 20
seconds.
    val ssc = new StreamingContext(sc, Seconds(20))

    println("Spark streaming context created")

    // Create a DStream that will connect to hostname:port,localhost:9999
    val lines = ssc.socketTextStream("localhost", 9999)

    // Split each line into words
    val words = lines.flatMap(_.split(" ")).map(x => x)

    // filter the offensive words from input string by using set and count the words
    val Offensive_Word_Count = words.filter(x => offensive_word_list.contains(x)).map(x
=> (x, 1)).reduceByKey(_ + _)

    Offensive_Word_Count.print()

    // Start the computation
    ssc.start()

    // Wait for the computation to terminate
    ssc.awaitTermination()

  }
}
```

After executing the program, the stream is initiated by entering data through the port 9999



```
[acadgild@localhost ~]$ nc -lk 9999
He is rambling like an idiot
you r an idiot
he is a bad person
she thinks that astrology is nonsense
do not talk nonsense to me
this is nonsense
```

The program reads the data as soon as every line is entered in the stream. The words are compared with the list of offensive words and whenever an offensive word is encountered, the count is incremented and the data is displayed on the console.



```
<terminated> Offensive_Words_Count$ [Scala Application] /usr/java/jdk1.8.0_151/bin/java (Jul 24, 2018, 7:09:02 PM)
-------------------------------------------
Time: 1532439580000 ms
-------------------------------------------
(idiot,2)

18/07/24 19:09:46 WARN RandomBlockReplicationPolicy: Expecting 1 replicas with only 0 peer/s.
18/07/24 19:09:46 WARN BlockManager: Block input-0-1532439586000 replicated to only 0 peer(s)
18/07/24 19:09:56 WARN RandomBlockReplicationPolicy: Expecting 1 replicas with only 0 peer/s.
18/07/24 19:09:56 WARN BlockManager: Block input-0-1532439596600 replicated to only 0 peer(s)
-------------------------------------------
Time: 1532439600000 ms
-------------------------------------------
(bad,1)
(nonsense,1)

18/07/24 19:10:12 WARN RandomBlockReplicationPolicy: Expecting 1 replicas with only 0 peer/s.
18/07/24 19:10:12 WARN BlockManager: Block input-0-1532439612200 replicated to only 0 peer(s)
18/07/24 19:10:17 WARN RandomBlockReplicationPolicy: Expecting 1 replicas with only 0 peer/s.
18/07/24 19:10:17 WARN BlockManager: Block input-0-1532439617000 replicated to only 0 peer(s)
-------------------------------------------
Time: 1532439620000 ms
-------------------------------------------
(nonsense,2)

-------------------------------------------
```

The first count is after the first 2 lines are entered in the stream.
The second count is after the next 2lines are entered in the stream.
The third count is after the final 2 line are entered in the stream.