ASSIGNMENT 16.1

Problem Statement:

Given a list of numbers - List[Int] (1, 2, 3, 4, 5, 6, 7, 8, 9, 10)

1. Find the sum of all numbers
2. Find the total elements in the list
3. Calculate the average of the numbers in the list
4. Find the sum of all the even numbers in the list
5. Find the total number of elements in the list divisible by both 5 and 3

Solution:

Let's create a spark context and then a convert the given list into a Spark parallelized RDD (Resilient Distributed Dataset). A **parallelized RDD** is a collection of distributed data elements and is a basic unit of data abstraction on which various map/reduce operations can be performed.

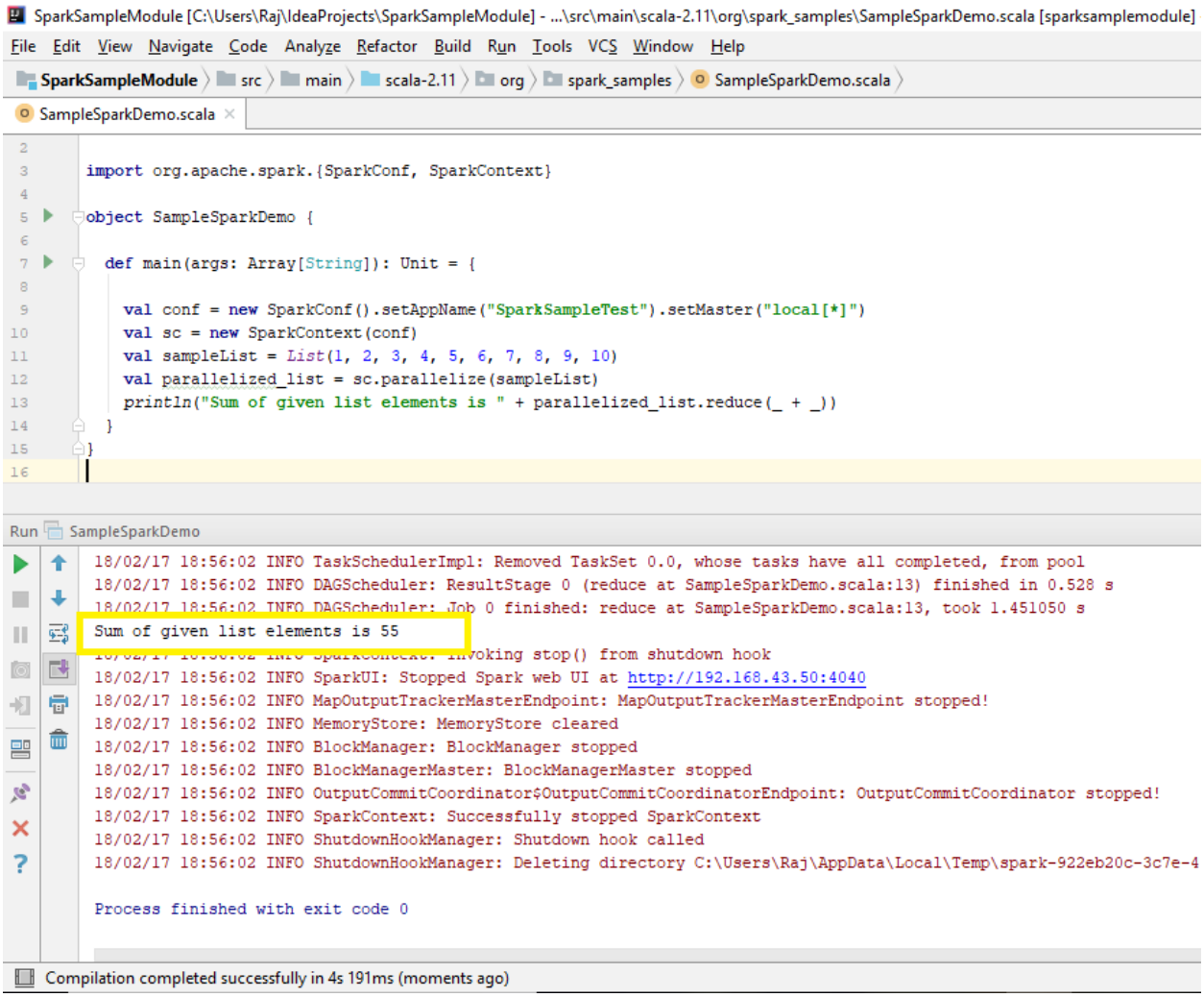
```
object SampleSparkDemo {

  def main(args: Array[String]): Unit = {

    val conf = new SparkConf().setAppName("SparkSampleTest").setMaster("local[*]")

                                        // 'conf' value is used to configure Spark environment

    val sc = new SparkContext(conf)        //'sc' is a Spark context variable created using SparkConf

    val sampleList = List(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)   //create and initialize a list with given values

    val parallelized_list = sc.parallelize(sampleList)   //create a parallelized RDD from sampleList

    }

}
```

1. Scala code to find the sum of all numbers:

```
println("Sum of given list elements is " + parallelized_list.reduce(_ + _))         // reduce() method
                        // is an action that takes two parameters of given RDD and applies '+' on them
```

Output:

Sum of given list elements is 55

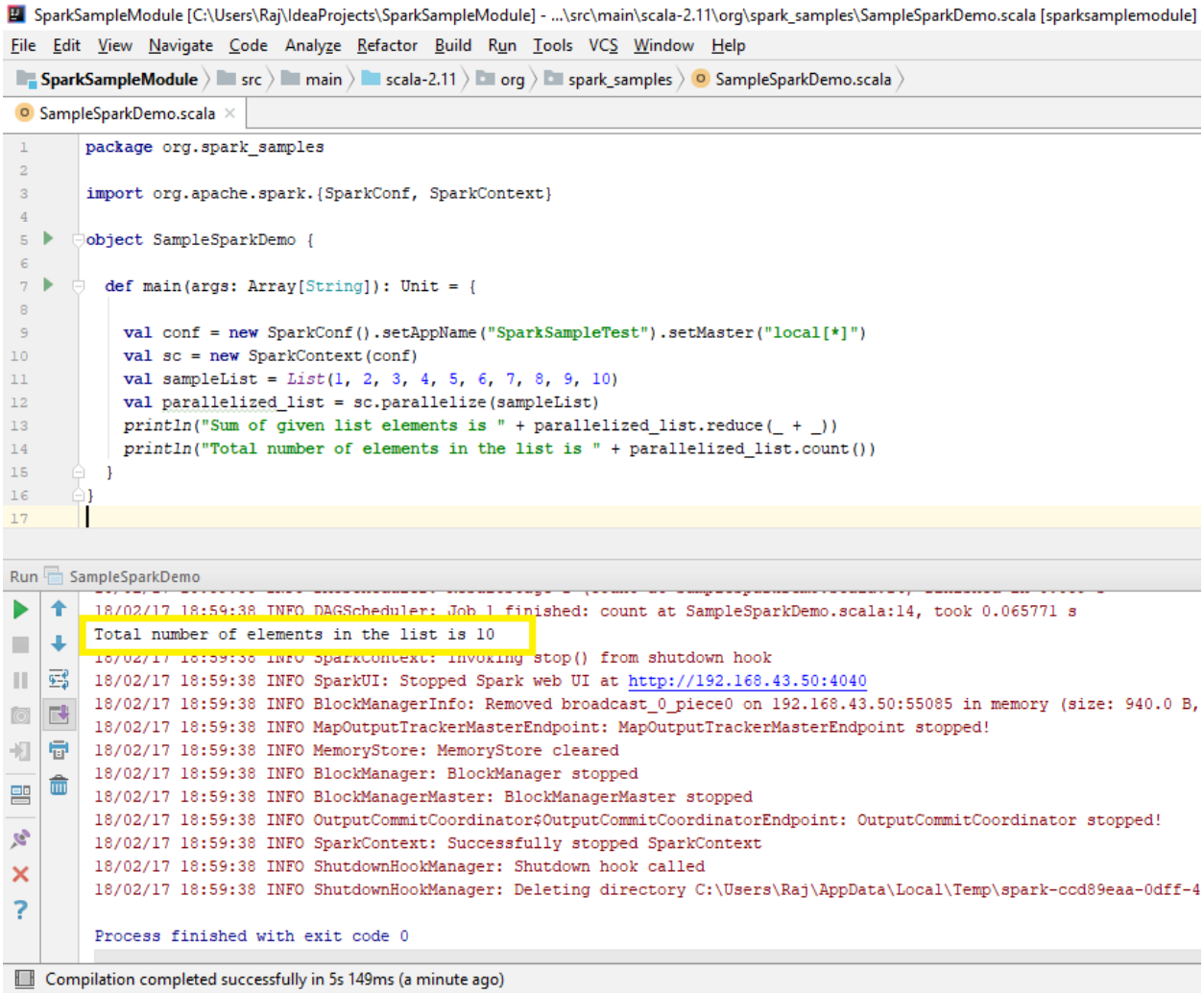


2. Scala code to find the total elements in the list:

```
println("Total number of elements in the list is " + parallelized_list.count())
                    // count() method is an action that returns count of elements in an input RDD
```

Output:

Total number of elements in the list is 10



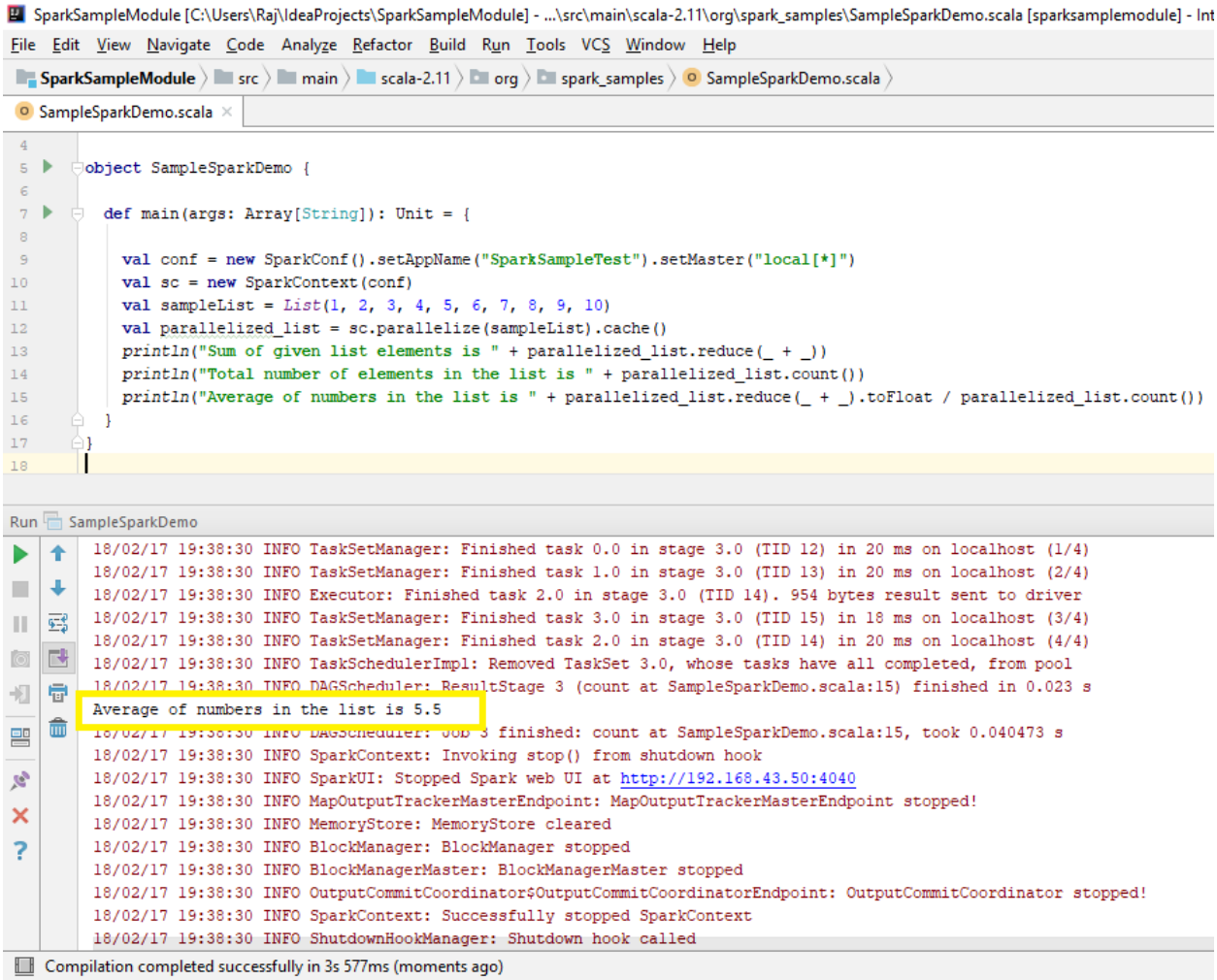
3. Scala code to calculate the average of the numbers in the list:

```
println("Average of numbers in the list is " + parallelized_list.reduce(_ + _).toFloat / parallelized_list.count())    //This is the calculates the sum via reduce(), gets number of elements
                                                                                                                        // in the RDD and finally divides the sum by number of items to get
                                                                                                                        // the average value as result
```

Note: The reason for including the computation inside println() statement is **to demonstrate the simplicity of coding in Spark.** To make it more understandable, we can separate each computational statement and print resulting value in the end.

Output:

Average of numbers in the list is 5.5



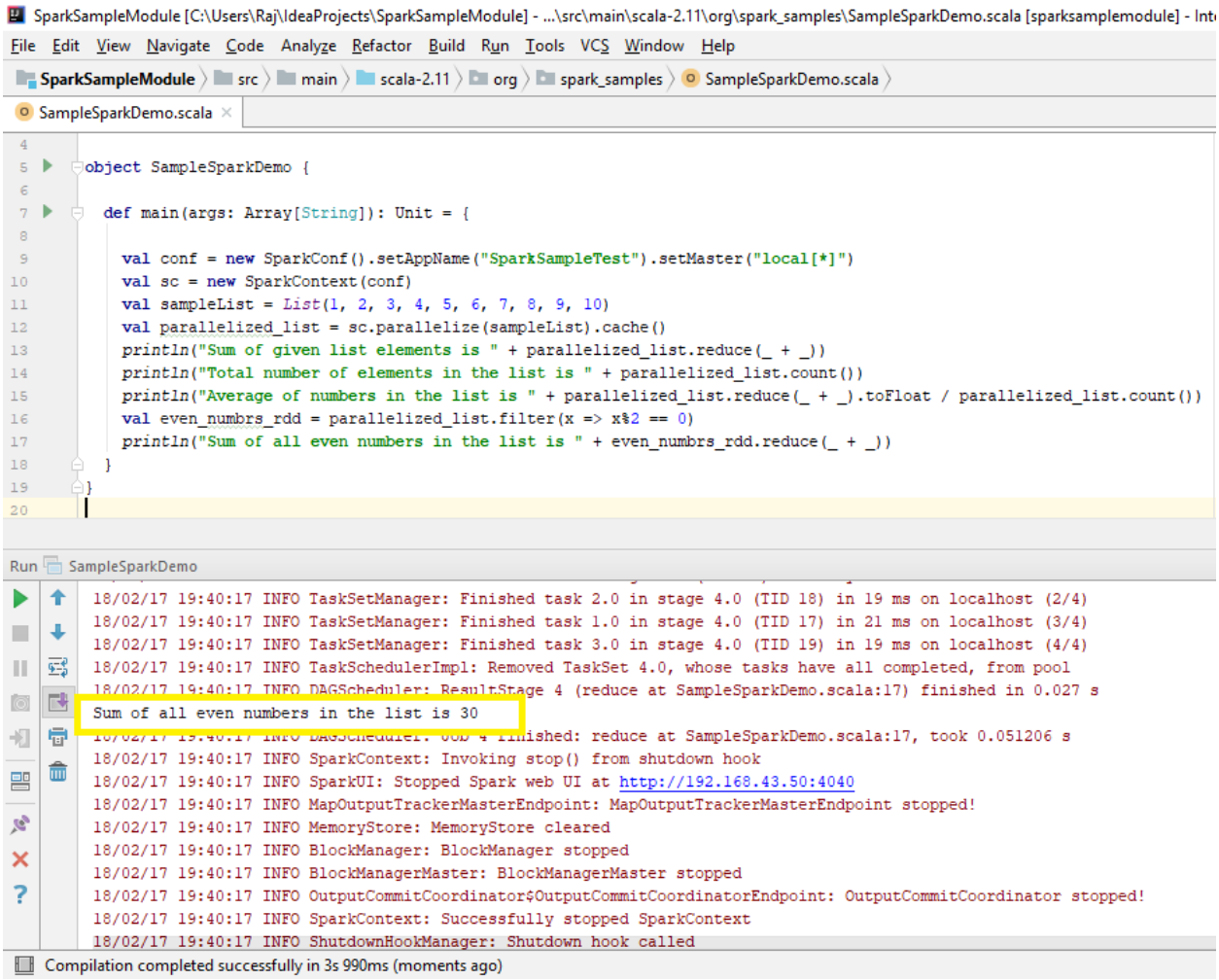
4. Scala code to find the sum of all the even numbers in the list:

val even_numbrs_rdd = parallelized_list.filter(x => x%2 == 0) //return an RDD of even numbers

println("Sum of all even numbers in the list is " + even_numbrs_rdd.reduce(_ + _)) // print sum

Output:

Sum of all even numbers in the list is 30



5. **Scala code to find the total number of elements in the list divisible by both 5 and 3:**

val divisible_by_5_and_3_rdd = parallelized_list.filter(x => (x % 5 == 0) && (x % 3 == 0))

println("Total number of elements divisible by both 5 and 3 is " + divisible_by_5_and_3_rdd.count())

The first statement applies filter() function to get only those numbers that can be divisible by both 5 and 3. The next statement prints the count of elements in that RDD.

Output:

Total number of elements divisible by both 5 and 3 is 0

Note: As there are no numbers in the given list which are divisible by both 5 and 3, we got the count as zero.

SparkSampleModule [C:\Users\Raj\IdeaProjects\SparkSampleModule] - ...\src\main\scala-2.11\org\spark_samples\SampleSparkDemo.scala [sparksamplemodule] - Int

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

SparkSampleModule > src > main > scala-2.11 > org > spark_samples > SampleSparkDemo.scala

SampleSparkDemo.scala

```scala
object SampleSparkDemo {

  def main(args: Array[String]): Unit = {

    val conf = new SparkConf().setAppName("SparkSampleTest").setMaster("local[*]")
    val sc = new SparkContext(conf)
    val sampleList = List(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
    val parallelized_list = sc.parallelize(sampleList).cache()
    println("Sum of given list elements is " + parallelized_list.reduce(_ + _))
    println("Total number of elements in the list is " + parallelized_list.count())
    println("Average of numbers in the list is " + parallelized_list.reduce(_ + _).toFloat / parallelized_list.count())
    val even_numbrs_rdd = parallelized_list.filter(x => x%2 == 0)
    println("Sum of all even numbers in the list is " + even_numbrs_rdd.reduce(_ + _))
    val divisible_by_5_and_3_rdd = parallelized_list.filter(x => (x % 5 == 0) && (x % 3 == 0))
    println("Total number of elements divisible by both 5 and 3 is " + divisible_by_5_and_3_rdd.count())
  }
}
```

Run SampleSparkDemo

```
18/02/17 22:31:54 INFO TaskSetManager: Finished task 0.0 in stage 5.0 (TID 20) in 19 ms on localhost (2/4)
18/02/17 22:31:54 INFO TaskSetManager: Finished task 2.0 in stage 5.0 (TID 22) in 19 ms on localhost (3/4)
18/02/17 22:31:54 INFO TaskSetManager: Finished task 1.0 in stage 5.0 (TID 21) in 26 ms on localhost (4/4)
18/02/17 22:31:54 INFO TaskSchedulerImpl: Removed TaskSet 5.0, whose tasks have all completed, from pool
Total number of elements divisible by both 5 and 3 is 0
18/02/17 22:31:54 INFO DAGScheduler: ResultStage 5 (count at SampleSparkDemo.scala:19) finished in 0.030 s
18/02/17 22:31:54 INFO DAGScheduler: Job 5 finished: count at SampleSparkDemo.scala:19, took 0.050415 s
18/02/17 22:31:54 INFO SparkContext: Invoking stop() from shutdown hook
18/02/17 22:31:54 INFO SparkUI: Stopped Spark web UI at http://192.168.43.50:4040
18/02/17 22:31:54 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
18/02/17 22:31:54 INFO MemoryStore: MemoryStore cleared
18/02/17 22:31:54 INFO BlockManager: BlockManager stopped
18/02/17 22:31:54 INFO BlockManagerMaster: BlockManagerMaster stopped
18/02/17 22:31:54 INFO OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputCommitCoordinator stopped!
18/02/17 22:31:54 INFO SparkContext: Successfully stopped SparkContext
```

Compilation completed successfully in 7s 239ms (a minute ago)

P. S.: If you have noticed the code in few of the screenshots above, I made use of cache() function while creating a parallelized collection. This function is useful when you want to perform the same operation repeatedly on an RDD. For example, we have applied count() on ‘parallelized_list’ RDD couple of times. In case we don’t use cache() on it, the count operation will read contents from source each time, whereas in the application of cache(), first time when count() is used, the data will be read from this RDD, cached and then counted. Second time when count() is used, the data will be read from cache and returns the count almost immediately. This function is recommended to be used while reading large amount of data from source so as to reduce data read time.