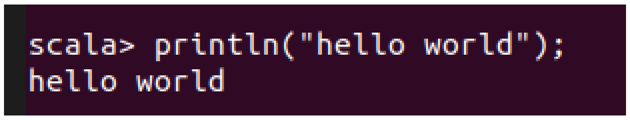
Program 1:

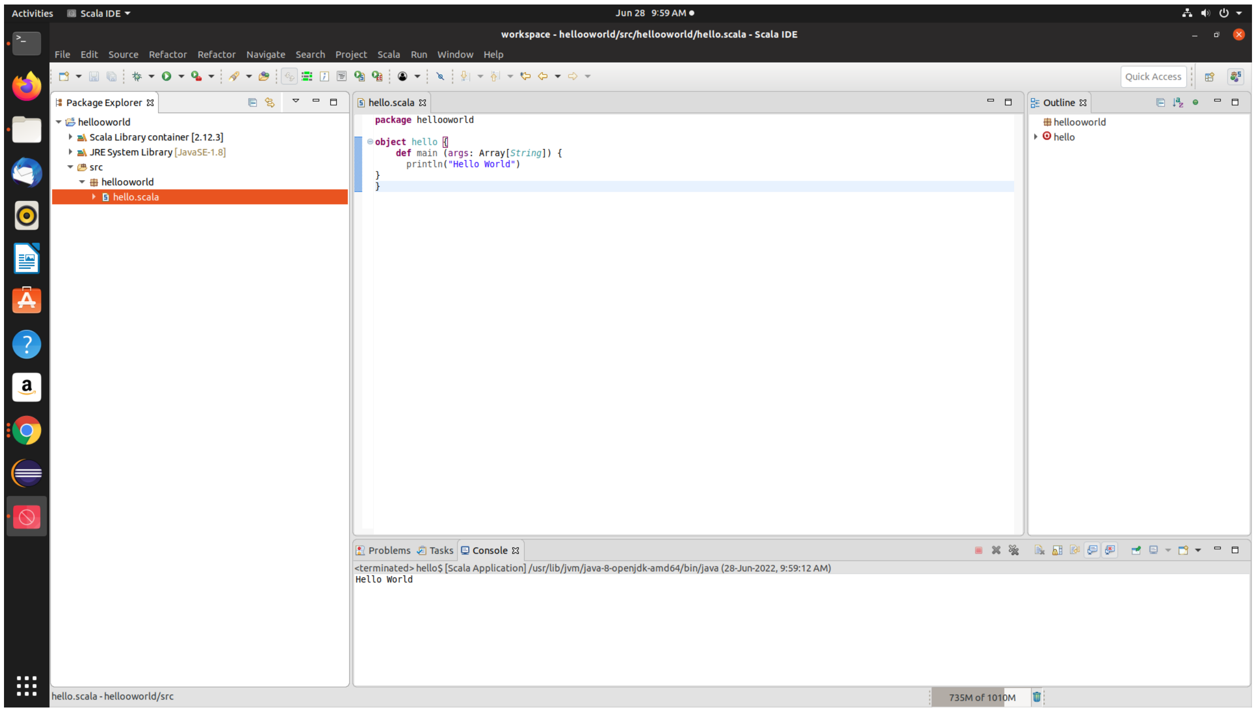
Print Hello Word at scala shell.



```
scala> println("hello world");
hello world
```

Program:2

1. Execute Hello World Program in SCALA IDE. Follow the steps given in



workspace - hellooworld/src/hellooworld/hello.scala - Scala IDE

hello.scala

```
package hellooworld

object hello {
    def main (args: Array[String]) {
      println("Hello World")
}
}
```

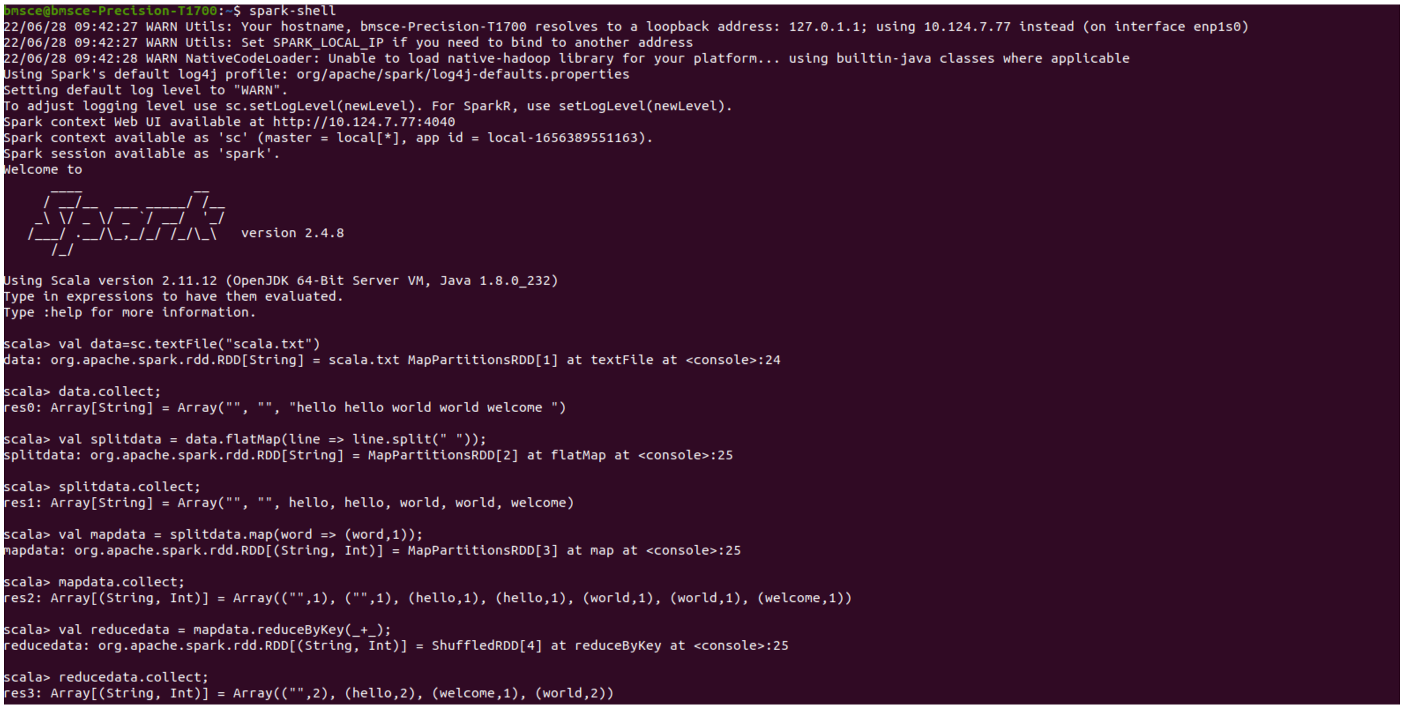
Console

```
<terminated> hello$ [Scala Application] /usr/lib/jvm/java-8-openjdk-amd64/bin/java (28-Jun-2022, 9:59:12 AM)
Hello World
```

Program :3

Program to run wordcount on scala shell

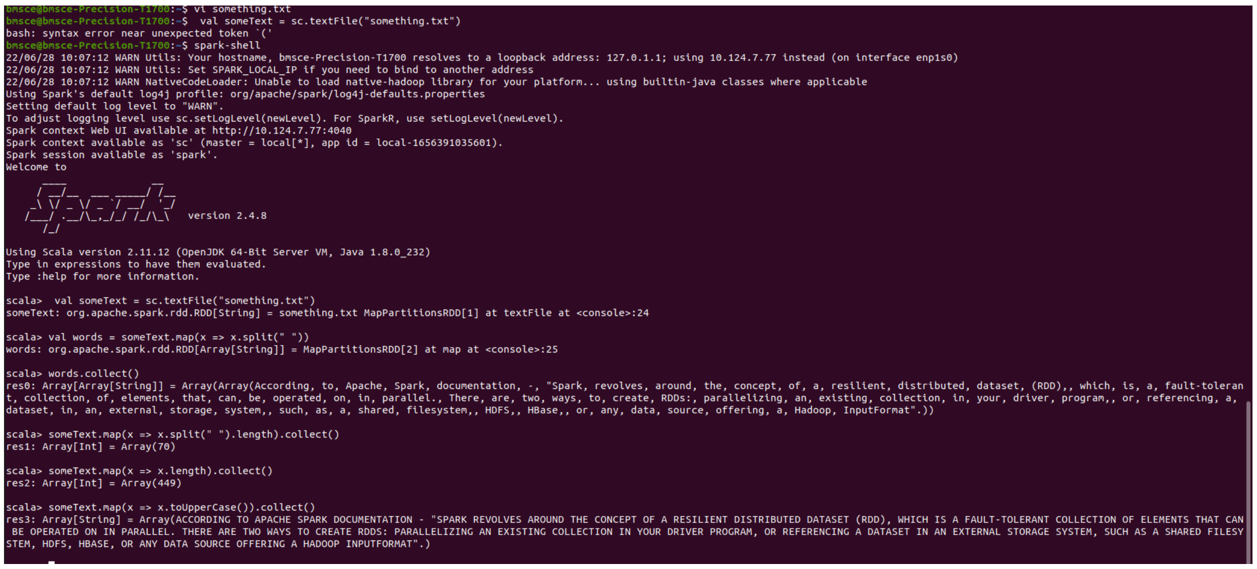
Note- Create a textfile sparkdata.txt locally and give appropriate path while loading the data using sc.textFile



```
bmsce@bmsce-Precision-T1700:~$ spark-shell
22/06/28 09:42:27 WARN Utils: Your hostname, bmsce-Precision-T1700 resolves to a loopback address: 127.0.1.1; using 10.124.7.77 instead (on interface enp1s0)
22/06/28 09:42:27 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another address
22/06/28 09:42:28 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
Spark context Web UI available at http://10.124.7.77:4040
Spark context available as 'sc' (master = local[*], app id = local-1656389551163).
Spark session available as 'spark'.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 2.4.8
      /_/

Using Scala version 2.11.12 (OpenJDK 64-Bit Server VM, Java 1.8.0_232)
Type in expressions to have them evaluated.
Type :help for more information.

scala> val data=sc.textFile("scala.txt")
data: org.apache.spark.rdd.RDD[String] = scala.txt MapPartitionsRDD[1] at textFile at <console>:24

scala> data.collect;
res0: Array[String] = Array("", "", "hello hello world world welcome ")

scala> val splitdata = data.flatMap(line => line.split(" "));
splitdata: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[2] at flatMap at <console>:25

scala> splitdata.collect;
res1: Array[String] = Array("", "", hello, hello, world, world, welcome)

scala> val mapdata = splitdata.map(word => (word,1));
mapdata: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[3] at map at <console>:25

scala> mapdata.collect;
res2: Array[(String, Int)] = Array(("",1), ("",1), (hello,1), (hello,1), (world,1), (world,1), (welcome,1))

scala> val reducedata = mapdata.reduceByKey(_+_);
reducedata: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[4] at reduceByKey at <console>:25

scala> reducedata.collect;
res3: Array[(String, Int)] = Array(("",2), (hello,2), (welcome,1), (world,2))
```

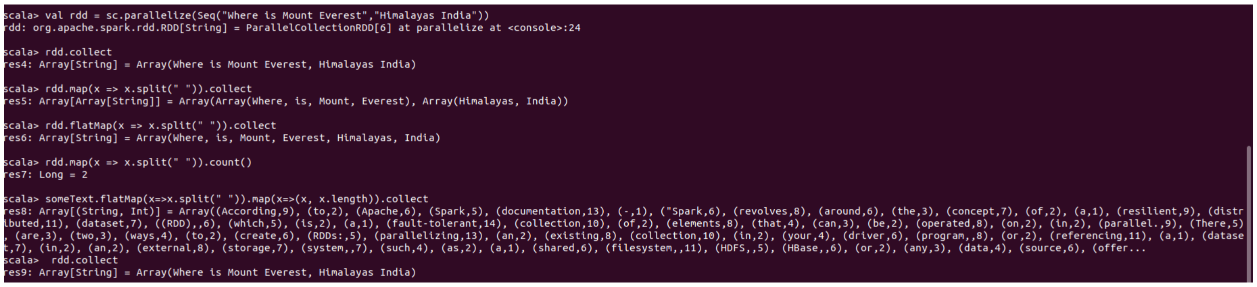
Spark Transformations

map(*func*):



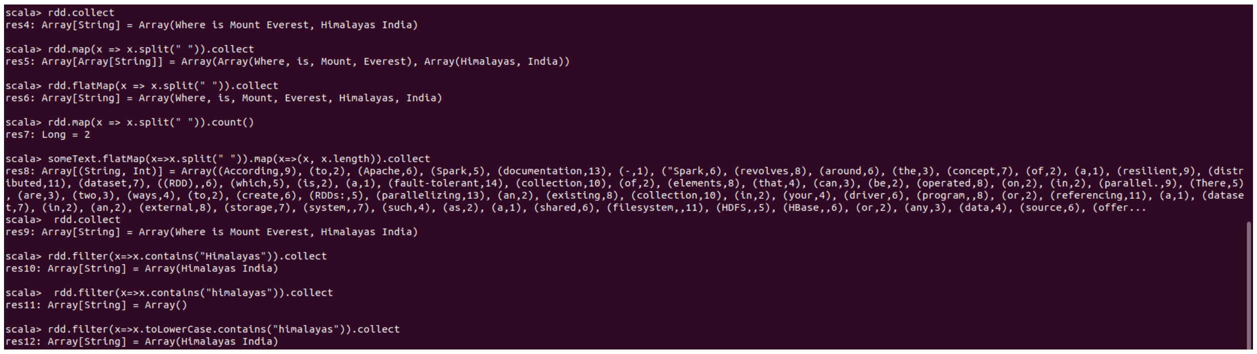
```
bmsce@bmsce-Precision-T1700:~$ vi something.txt
bmsce@bmsce-Precision-T1700:~$  val someText = sc.textFile("something.txt")
bash: syntax error near unexpected token `('
bmsce@bmsce-Precision-T1700:~$ spark-shell
22/06/28 10:07:12 WARN Utils: Your hostname, bmsce-Precision-T1700 resolves to a loopback address: 127.0.1.1; using 10.124.7.77 instead (on interface enp1s0)
22/06/28 10:07:12 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another address
22/06/28 10:07:12 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
Spark context Web UI available at http://10.124.7.77:4040
Spark context available as 'sc' (master = local[*], app id = local-1656391035601).
Spark session available as 'spark'.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 2.4.8
      /_/

Using Scala version 2.11.12 (OpenJDK 64-Bit Server VM, Java 1.8.0_232)
Type in expressions to have them evaluated.
Type :help for more information.

scala>  val someText = sc.textFile("something.txt")
someText: org.apache.spark.rdd.RDD[String] = something.txt MapPartitionsRDD[1] at textFile at <console>:24

scala> val words = someText.map(x => x.split(" "))
words: org.apache.spark.rdd.RDD[Array[String]] = MapPartitionsRDD[2] at map at <console>:25

scala> words.collect()
res0: Array[Array[String]] = Array(Array(According, to, Apache, Spark, documentation, -, "Spark, revolves, around, the, concept, of, a, resilient, distributed, dataset, (RDD),, which, is, a, fault-toleran
t, collection, of, elements, that, can, be, operated, on, in, parallel., There, are, two, ways, to, create, RDDs:, parallelizing, an, existing, collection, in, your, driver, program,, or, referencing, a,
dataset, in, an, external, storage, system,, such, as, a, shared, filesystem,, HDFS,, HBase,, or, any, data, source, offering, a, Hadoop, InputFormat".))

scala> someText.map(x => x.split(" ").length).collect()
res1: Array[Int] = Array(70)

scala> someText.map(x => x.length).collect()
res2: Array[Int] = Array(449)

scala> someText.map(x => x.toUpperCase()).collect()
res3: Array[String] = Array(ACCORDING TO APACHE SPARK DOCUMENTATION - "SPARK REVOLVES AROUND THE CONCEPT OF A RESILIENT DISTRIBUTED DATASET (RDD), WHICH IS A FAULT-TOLERANT COLLECTION OF ELEMENTS THAT CAN
 BE OPERATED ON IN PARALLEL. THERE ARE TWO WAYS TO CREATE RDDS: PARALLELIZING AN EXISTING COLLECTION IN YOUR DRIVER PROGRAM, OR REFERENCING A DATASET IN AN EXTERNAL STORAGE SYSTEM, SUCH AS A SHARED FILESY
STEM, HDFS, HBASE, OR ANY DATA SOURCE OFFERING A HADOOP INPUTFORMAT".)
```

flatmap(*func*)

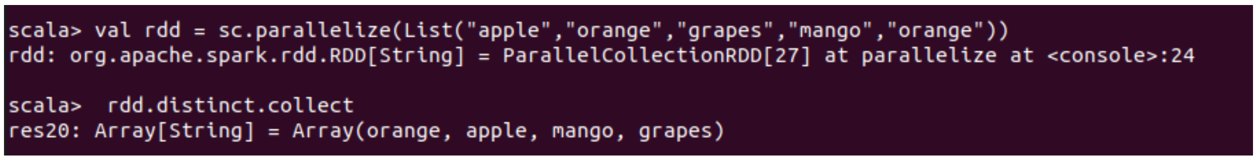


```
scala> val rdd = sc.parallelize(Seq("Where is Mount Everest","Himalayas India"))
rdd: org.apache.spark.rdd.RDD[String] = ParallelCollectionRDD[6] at parallelize at <console>:24

scala> rdd.collect
res4: Array[String] = Array(Where is Mount Everest, Himalayas India)

scala> rdd.map(x => x.split(" ")).collect
res5: Array[Array[String]] = Array(Array(Where, is, Mount, Everest), Array(Himalayas, India))

scala> rdd.flatMap(x => x.split(" ")).collect
res6: Array[String] = Array(Where, is, Mount, Everest, Himalayas, India)

scala> rdd.map(x => x.split(" ")).count()
res7: Long = 2

scala> someText.flatMap(x=>x.split(" ")).map(x=>(x, x.length)).collect
res8: Array[(String, Int)] = Array((According,9), (to,2), (Apache,6), (Spark,5), (documentation,13), (-,1), ("Spark,6), (revolves,8), (around,6), (the,3), (concept,7), (of,2), (a,1), (resilient,9), (distr
ibuted,11), (dataset,7), ((RDD),,6), (which,5), (is,2), (a,1), (fault-tolerant,14), (collection,10), (of,2), (elements,8), (that,4), (can,3), (be,2), (operated,8), (on,2), (in,2), (parallel.,9), (There,5)
, (are,3), (two,3), (ways,4), (to,2), (create,6), (RDDs:,5), (parallelizing,13), (an,2), (existing,8), (collection,10), (in,2), (your,4), (driver,6), (program,,8), (or,2), (referencing,11), (a,1), (datase
t,7), (in,2), (an,2), (external,8), (storage,7), (system,,7), (such,4), (as,2), (a,1), (shared,6), (filesystem,,11), (HDFS,,5), (HBase,,6), (or,2), (any,3), (data,4), (source,6), (offer...
scala>  rdd.collect
res9: Array[String] = Array(Where is Mount Everest, Himalayas India)
```

filter(*func*)



```
scala> rdd.collect
res4: Array[String] = Array(Where is Mount Everest, Himalayas India)

scala> rdd.map(x => x.split(" ")).collect
res5: Array[Array[String]] = Array(Array(Where, is, Mount, Everest), Array(Himalayas, India))

scala> rdd.flatMap(x => x.split(" ")).collect
res6: Array[String] = Array(Where, is, Mount, Everest, Himalayas, India)

scala> rdd.map(x => x.split(" ")).count()
res7: Long = 2

scala> someText.flatMap(x=>x.split(" ")).map(x=>(x, x.length)).collect
res8: Array[(String, Int)] = Array((According,9), (to,2), (Apache,6), (Spark,5), (documentation,13), (-,1), ("Spark,6), (revolves,8), (around,6), (the,3), (concept,7), (of,2), (a,1), (resilient,9), (distr
ibuted,11), (dataset,7), ((RDD),,6), (which,5), (is,2), (a,1), (fault-tolerant,14), (collection,10), (of,2), (elements,8), (that,4), (can,3), (be,2), (operated,8), (on,2), (in,2), (parallel.,9), (There,5)
, (are,3), (two,3), (ways,4), (to,2), (create,6), (RDDs:,5), (parallelizing,13), (an,2), (existing,8), (collection,10), (in,2), (your,4), (driver,6), (program,,8), (or,2), (referencing,11), (a,1), (datase
t,7), (in,2), (an,2), (external,8), (storage,7), (system,,7), (such,4), (as,2), (a,1), (shared,6), (filesystem,,11), (HDFS,,5), (HBase,,6), (or,2), (any,3), (data,4), (source,6), (offer...
scala>  rdd.collect
res9: Array[String] = Array(Where is Mount Everest, Himalayas India)

scala> rdd.filter(x=>x.contains("Himalayas")).collect
res10: Array[String] = Array(Himalayas India)

scala>  rdd.filter(x=>x.contains("himalayas")).collect
res11: Array[String] = Array()

scala> rdd.filter(x=>x.toLowerCase.contains("himalayas")).collect
res12: Array[String] = Array(Himalayas India)
```

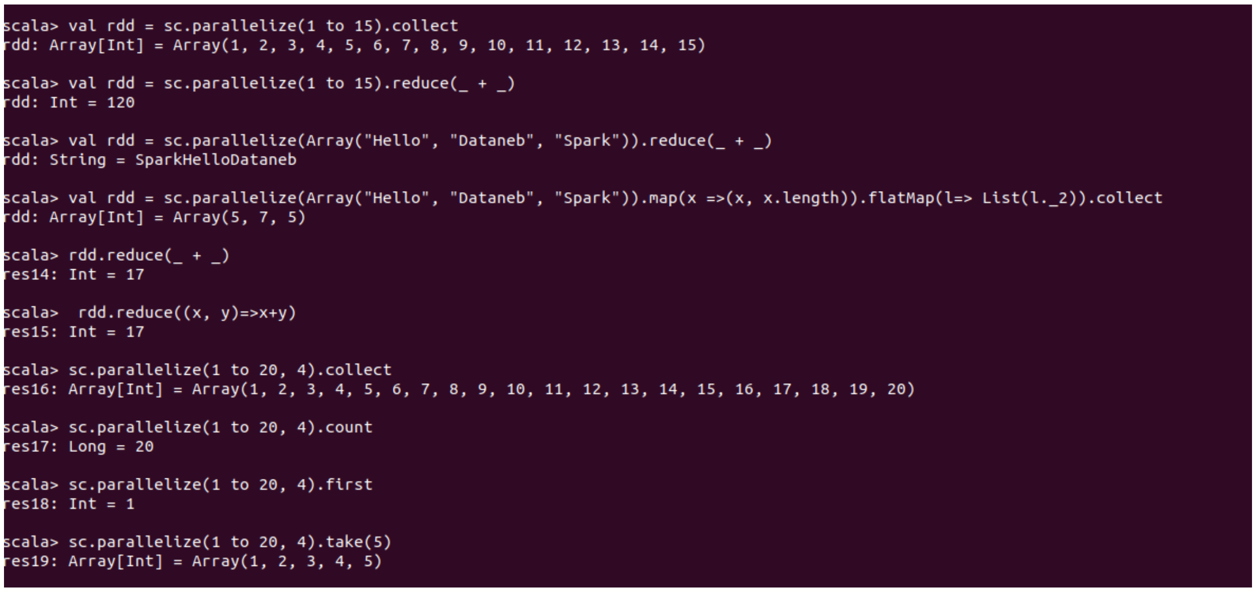


```
scala> val rdd = sc.parallelize(List("apple","orange","grapes","mango","orange"))
rdd: org.apache.spark.rdd.RDD[String] = ParallelCollectionRDD[27] at parallelize at <console>:24

scala>  rdd.distinct.collect
res20: Array[String] = Array(orange, apple, mango, grapes)
```

Spark Actions

reduce()

collect(), count(), first(), take()



```
scala> val rdd = sc.parallelize(1 to 15).collect
rdd: Array[Int] = Array(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15)

scala> val rdd = sc.parallelize(1 to 15).reduce(_ + _)
rdd: Int = 120

scala> val rdd = sc.parallelize(Array("Hello", "Dataneb", "Spark")).reduce(_ + _)
rdd: String = SparkHelloDataneb

scala> val rdd = sc.parallelize(Array("Hello", "Dataneb", "Spark")).map(x =>(x, x.length)).flatMap(l=> List(l._2)).collect
rdd: Array[Int] = Array(5, 7, 5)

scala> rdd.reduce(_ + _)
res14: Int = 17

scala>  rdd.reduce((x, y)=>x+y)
res15: Int = 17

scala> sc.parallelize(1 to 20, 4).collect
res16: Array[Int] = Array(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20)

scala> sc.parallelize(1 to 20, 4).count
res17: Long = 20

scala> sc.parallelize(1 to 20, 4).first
res18: Int = 1

scala> sc.parallelize(1 to 20, 4).take(5)
res19: Array[Int] = Array(1, 2, 3, 4, 5)
```