BDA Assignment 4

Name: Prathibha R
USN: 1BM18CS074

---

1. map()

scala> val input = sc.parallelize(List(1,2,3,4))
input: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[0] at parallelize at <console>:24

scala> val result = input.map(x => x*x)
result: org.apache.spark.rdd.RDD[Int] = MapPartitionsRDD[1] at map at <console>:25

scala> println(result.collect().mkString(","))
1,4,9,16

2. flatmap()

scala> val lines = sc.parallelize(List("hello world","hi"))

lines: org.apache.spark.rdd.RDD[String] = ParallelCollectionRDD[2] at parallelize at <console>:24
scala> val words = lines.flatMap(line => line.split(""))
words: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[3] at flatMap at <console>:25

scala> words.first()
res1: String = h

3. filter()

```
scala> val input = sc.parallelize(List(1,2,3,4))
input: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[0] at parallelize
at <console>:24

scala> val result = input.filter(x => x!= 1)
result: org.apache.spark.rdd.RDD[Int] = MapPartitionsRDD[1] at filter at <conso
le>:25

scala> println(result.collect().mkString(","))
[Stage 0:>                                                          (0 + 0) / 2
[Stage 0:>                                                          (0 + 2) / 2

2,3,4
```

scala> val input = sc.parallelize(List(1,2,3,4))
input: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[0] at parallelize at <console>:24

scala> val result = input.filter(x => x!= 1)
result: org.apache.spark.rdd.RDD[Int] = MapPartitionsRDD[1] at filter at <console>:25

scala> println(result.collect().mkString(","))
2,3,4

4. distinct()

```
scala> val input4 = sc.parallelize(List(1,2,2,3,3,4))
input4: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[2] at parallelize
 at <console>:24

scala> val result = input4.distinct()
result: org.apache.spark.rdd.RDD[Int] = MapPartitionsRDD[5] at distinct at <con
sole>:25

scala> println(result.collect().mkString(","))
4,2,1,3
```

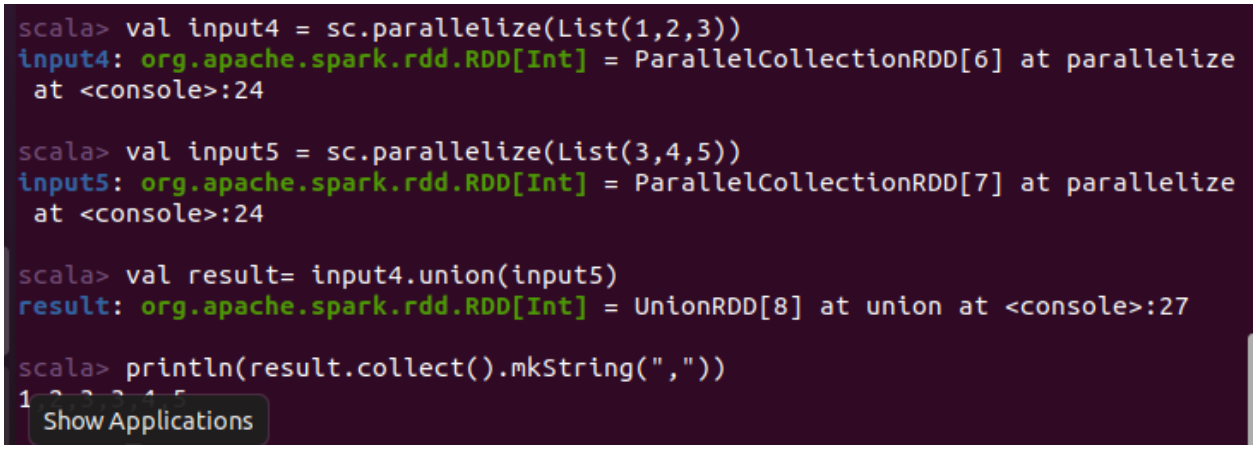
scala> val input4 = sc.parallelize(List(1,2,2,3,3,4))

input4: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[2] at parallelize at <console>:24

scala> val result = input4.distinct()
result: org.apache.spark.rdd.RDD[Int] = MapPartitionsRDD[5] at distinct at <console>:25

scala> println(result.collect().mkString(","))
4,2,1,3

5. union()



scala> val input4 = sc.parallelize(List(1,2,3))
input4: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[6] at parallelize at <console>:24

scala> val input5 = sc.parallelize(List(3,4,5))
input5: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[7] at parallelize at <console>:24

scala> val result= input4.union(input5)
result: org.apache.spark.rdd.RDD[Int] = UnionRDD[8] at union at <console>:27

scala> println(result.collect().mkString(","))
1,2,3,3,4,5

6. intersection()

```
scala> val input4 = sc.parallelize(List(1,2,3))
input4: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[0] at parallelize at <console>:24

scala> val input5 = sc.parallelize(List(3,4,5))
input5: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[1] at parallelize at <console>:24

scala> val result= input4.intersection(input5)
result: org.apache.spark.rdd.RDD[Int] = MapPartitionsRDD[7] at intersection at <console>:27

scala> println(result.collect().mkString(","))
[Stage 0:>                                                          (0 + 1) / 2[Stage 0:>
                            (0 + 2) / 2[Stage 0:>                    (0 + 2) / 2][Stage 1:>                  (0 + 0)
/ 2[Stage 0:>                (0 + 2) / 2][Stage 1:>                    (0 + 2) / 2[Stage 2:>
                               (0 + 2) / 2
      3
```

scala> val input4 = sc.parallelize(List(1,2,3))
input4: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[8] at parallelize at :24

scala> val input5 = sc.parallelize(List(3,4,5))
input5: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[9] at parallelize at :24

scala> val result = input4.intersection(input5)
result: org.apache.spark.rdd.RDD[Int] = MapPartitionsRDD[16] at intersection at :27

scala> println(result.collect().mkString(","))
3

7. subtract()

```
scala> val input4 = sc.parallelize(List(1,2,3))
input4: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[8] at parallelize at <console>:24

scala> val input5 = sc.parallelize(List(3,4,5))
input5: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[9] at parallelize at <console>:24

scala> val result= input4.subtract(input5)
result: org.apache.spark.rdd.RDD[Int] = MapPartitionsRDD[13] at subtract at <console>:27

scala> println(result.collect().mkString(","))
2,1
```

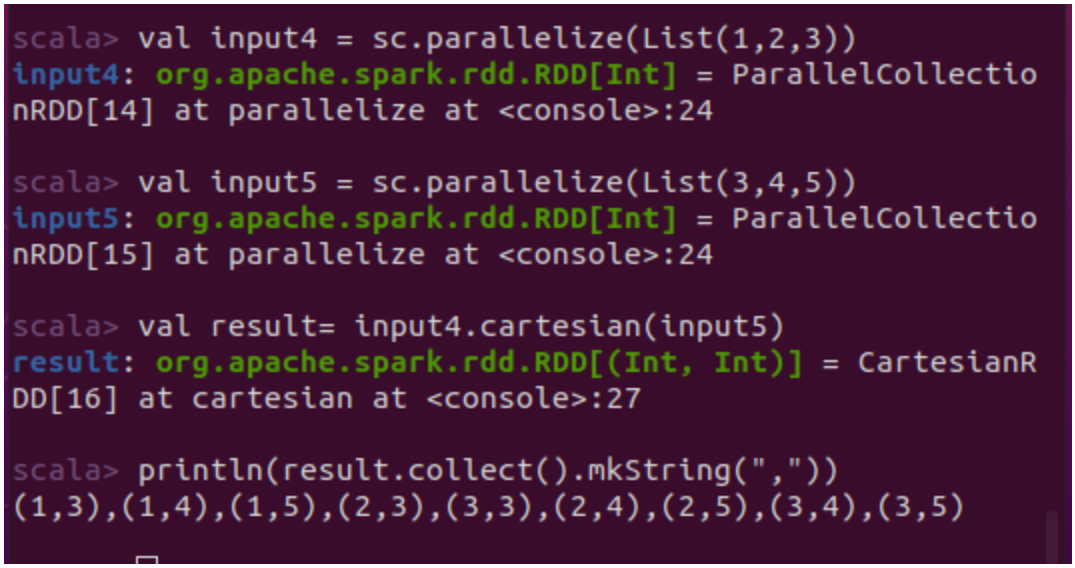
scala> val input4 = sc.parallelize(List(1,2,3))
input4: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[8] at parallelize at :24

scala> val input5 = sc.parallelize(List(3,4,5))
input5: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[9] at parallelize at :24

scala> val result = input4.subtract(input5)
result: org.apache.spark.rdd.RDD[Int] = MapPartitionsRDD[20] at subtract at :27

scala> println(result.collect().mkString(","))
2,1

8. cartesian()



scala> val input4 = sc.parallelize(List(1,2,3))
input4: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[8] at parallelize at :24

scala> val input5 = sc.parallelize(List(3,4,5))
input5: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[9] at parallelize at :24

scala> val result = input4.cartesian(input5)
result: org.apache.spark.rdd.RDD[(Int, Int)] = CartesianRDD[21] at cartesian at :27

scala> println(result.collect().mkString(","))
(1,3),(1,4),(1,5),(2,3),(2,4),(2,5),(3,3),(3,4),(3,5)