Task 1

Given a list of numbers - List[Int] (1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
- find the sum of all numbers
- find the total elements in the list
- calculate the average of the numbers in the list
- find the sum of all the even numbers in the list
- find the total number of elements in the list divisible by both 5 and 3

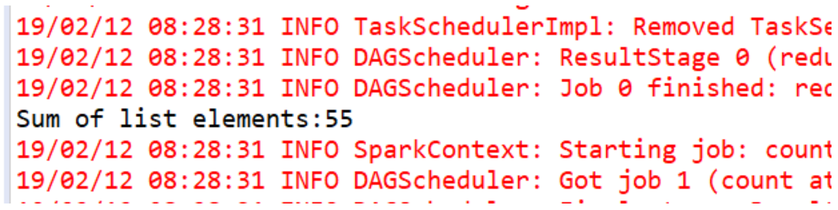
Code-

```
import org.apache.spark.SparkConf
import org.apache.spark.SparkContext

object Acadgild_spark1 {
  def main(args:Array[String]){
    val conf=new SparkConf().setAppName("Acadgild").setMaster("local")
    val sc=new SparkContext(conf)
    System.setProperty("hadoop.home.dir", "C://winutils")
    val num_list=List(1,2,3,4,5,6,7,8,9,10)
    val rdd_list=sc.parallelize(num_list)
    //sum of all numbers
    println("Sum of list elements:"+rdd_list.reduce((a,b)=>a+b))
    //find the total elements in the list
    println("Total elements in list:"+rdd_list.count())
    println("Average of number in
list:"+rdd_list.reduce((a,b)=>a+b)/rdd_list.count())
    //find the sum of all the even numbers in the list
    val even_list=rdd_list.map(x=> if(x%2==0) x else 0)
    println("Sum of even numbers:"+even_list.sum())
    //find the total number of elements in the list divisible by both 5 and 3
    val div_list=rdd_list.filter(x=> x%5==0 && x%3==0)
    println("Elements divisible by both 5 and 3:"+div_list.count())
  }
}
```

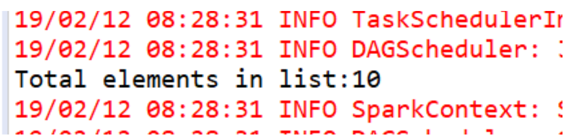
Output-

find the sum of all numbers



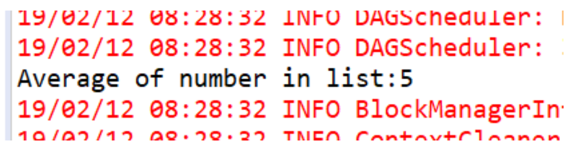
```
19/02/12 08:28:31 INFO TaskSchedulerImpl: Removed TaskSe
19/02/12 08:28:31 INFO DAGScheduler: ResultStage 0 (redu
19/02/12 08:28:31 INFO DAGScheduler: Job 0 finished: rec
Sum of list elements:55
19/02/12 08:28:31 INFO SparkContext: Starting job: count
19/02/12 08:28:31 INFO DAGScheduler: Got job 1 (count at
```

find the total elements in the list



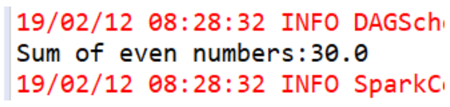
```
19/02/12 08:28:31 INFO TaskSchedulerIr
19/02/12 08:28:31 INFO DAGScheduler: :
Total elements in list:10
19/02/12 08:28:31 INFO SparkContext: :
```

calculate the average of the numbers in the list



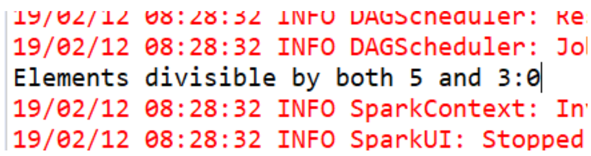
```
19/02/12 08:28:32 INFO DAGScheduler:
19/02/12 08:28:32 INFO DAGScheduler:
Average of number in list:5
19/02/12 08:28:32 INFO BlockManagerIn
19/02/12 08:28:32 INFO ContextCleaner
```

find the sum of all the even numbers in the list



```
19/02/12 08:28:32 INFO DAGSch
Sum of even numbers:30.0
19/02/12 08:28:32 INFO SparkC
```

find the total number of elements in the list divisible by both 5 and 3



```
19/02/12 08:28:32 INFO DAGScheduler: Re
19/02/12 08:28:32 INFO DAGScheduler: Jo
Elements divisible by both 5 and 3:0
19/02/12 08:28:32 INFO SparkContext: In
19/02/12 08:28:32 INFO SparkUI: Stopped
```

Task 2

1) Pen down the limitations of MapReduce.

While development of Hadoop's MapReduce the vision was pretty limited, it was developed just to handle 1 problem: Batch processing.

MapReduce cannot handle:

1. Interactive Processing
2. Real-time (stream) Processing
3. Iterative (delta) Processing
4. In-memory Processing
5. Graph Processing

2) What is RDD? Explain few features of RDD?

RDD stands for "Resilient Distributed Dataset". It is the fundamental data structure of Apache Spark. RDD in Apache Spark is an immutable collection of objects which computes on the different node of the cluster.
Decomposing the name RDD:

- Resilient, i.e. fault-tolerant with the help of RDD lineage graph(DAG) and so able to recompute missing or damaged partitions due to node failures.
- Distributed, since Data resides on multiple nodes.
- Dataset represents records of the data you work with. The user can load the data set externally which can be either JSON file, CSV file, text file or database via JDBC with no specific data structure.

Hence, each and every dataset in RDD is logically partitioned across many servers so that they can be computed on different nodes of the cluster. RDDs are fault tolerant i.e. It posses self-recovery in the case of failure.

3) List down few Spark RDD operations and explain each of them.

Two types of **Apache Spark** RDD operations are- Transformations and Actions.
A **Transformation** is a function that produces new **RDD** from the existing RDDs but when we want to work with the actual dataset, at that point **Action** is performed. When the action is triggered after the result, new RDD is not formed like transformation. In this Apache Spark RDD operations tutorial we will get the detailed view of what is Spark RDD, what is the transformation in Spark RDD, various RDD transformation operations in Spark with examples, what is action in Spark RDD and various RDD action operations in Spark with examples.

Apache Spark RDD supports two types of Operations-

- Transformations
- Actions
  - Spark Transformation is a function that produces new RDD from the existing RDDs. It takes RDD as input and produces one or more RDD as output. Each time it creates new RDD when we apply any transformation. Thus, the so input RDDs, cannot be changed since RDD are immutable in nature.
  - Applying transformation built an RDD lineage, with the entire parent RDDs of the final RDD(s). RDD lineage, also known as RDD operator graph or RDD dependency graph. It is a logical execution plan i.e., it is Directed Acyclic Graph (DAG) of the entire parent RDDs of RDD.
  - Transformations are lazy in nature i.e., they get execute when we call an action. They are not executed immediately. Two most basic type of transformations is a map(), filter().
    After the transformation, the resultant RDD is always different from its parent RDD. It can be smaller (e.g. filter, count, distinct, sample), bigger (e.g. flatMap(), union(), Cartesian()) or the same size (e.g. map).