2)
a) Write and execute a map reduce program to find whether a given number even or not.

→ Driver.java :-

```
import org.apache.hadoop.*;
public class Driver extends configured implements Tool
{
public int run(String args[]) throws Exception
  {
   if (args.length < 2){
      System.out.println("Error");
      return -1;
    }
   Jobconf conf = new Jobconf (Driver.class);
   FileInputFormat.setInputpaths(conf, new(args[0]));
   FileOutputFormat.SetOutpath(conf, new(args[1]));
   conf.SetMapperClass (EO.Mapper.class);
   conf.setReducerClass (EO.Reducer.class);
   conf.setMapOutputValueclass (IntWritable.Class);
   conf.setMapOutputKeyClass (Text class);
   conf.setOutputkey Class (Text class);
   conf.setOutputValueClass (IntWritable class);
   JobClient.runJob(conf);
    return 0;
}
```

```
public static void main(String args[]) throws
                                    Exception
{
  int exitcode = ToolRunner.run(new Driver(), args);
  System.out.println(exitcode);
}
}
```

→ EOMapper.java :-

```
import java.io.IOException;
import org.apache.hadoop.*;
public class EOMapper extends MapreduceBase
      Implements Mapper<long Writable, Text, Text,
                                    IntWritable>
{
  public void map(LongWritable key, Text value,
      outputCollector<Text, IntWritable> Output,
        Reporter rep) throws IOException
  {
    String data[] = value.toString().split(" ");
    for (String num : data)
    {
      int number = Integer.parseInt(num);
      if (number % 2 == 1) {
        output.collect(new Text("ODD"), new IntWritable
                                    (number));
      }else{
        output.collect(new Text("EVEN"), new IntWrita-
                                    ble(number));
      }
    }
  }
}
```

Nitish.N.B
1BM18CS065

→ EOReducer.java:-

```
import java.io.*;
import java.util.*;
import org.apache.hadoop.*;
public class EOReducer extends Mapreduce Base
implements Reducer < Text, IntWritable, Text,
                                    IntWritable >
{
  public void reduce (Text key, Iterator<IntWritable
      value, Output collector< Text, IntWritable>
      output, Reporter rep) throws Exception
  {
    int num;
    while(value.hasNext())
    {
      IntWritable i= value.Nesct();
      num = i.get();
      output.collect(key, new IntWritable(num));
    }
  }
}
```

b) Demonstrate 4 transformations & 4 actions on an RDD of your choice

* Transformations :-

```
val input = sc.parallelize(List(1,2,3,4))
```

1) Map() :-

```
val result = input.map(x => x * x)
println(result.collect())
```

2) Union() :-

```
val input1 = sc.parallelize(List(1,2,3,4))
val input2 = sc.parallelize(List(3,4,5,6))
val result = input1.union(input2)
println(result.collect())
```

3) Intersection :-

```
val input1 = sc.parallelize(List(1,2,3,4))
val input2 = sc.parallelize(List(1,2,3))
val result = input1.intersection(input2)
println(result.collect())
```

4) Distinct :-

```
val input1 = sc.parallelize(List(1,2,3,4,2,4))
val result = input.distinct()
println(result.collect())
```

Nitish.N.B
1BM18CS065

Actions :-

i) **Collect :-**

```
sc.parallelize(1 to 20, 4).collect()
```

ii) **Count**

```
sc.parallelize(1 to 20, 4).count()
```

iii) **first :-**

```
sc.parallelize(1 to 20, 4).first()
```

iv) **take(num) :-**

```
sc.parallelize(1 to 20, 4).take(5)
```


Nitish