20/5/25

PAGE :
DATE : / /

lab 9 :-

Spark :-

Write a Scala program to print numbers from 1 to 100 using for loop.

```
> spark-shell
scala> for (i <- 1 to 100){
          println(i)
       }
1
2
3
.
.
.
99
100
```

Using RDD and FlatMap ~~counter~~ count how many times each word appears in a file & write out a list of words whose count is strictly greater than 4 Using spark.

```
> nano input.txt
Hi Hi Hi Hi Hi Hi
Hello Hello Hello Hello Hello
is is is my world
```

```
> spark-shell
> val fileRDD = sc.textFile("input.txt")
> val wordsRDD = fileRDD.flatMap(line => line.split("\\W+")).filter(_.nonEmpty)
```

PAGE :
DATE : / /

> val wordCountsRDD = wordsRDD.map(word =>
(word.toLowerCase(), 1)).reduceByKey(_ + _)

> wordCountsRDD.collect().foreach(println)
(Hi, 6)
(Hello, 5)
(is, 3)
(my, 1)
(world, 1)

> val filteredWordsRDD = wordCountsRDD
filter{ case (word, count) => count > 4}

> val result = filteredWordsRDD.collect()

> result.foreach(println)
(Hi, 6)
(Hello, 5)

Q. For a given text file, create a map reduce program to sort the content in an alphabetical order ~~listing~~ only top 10 maximum Occurences of ~~words~~.

Mapper:

```
public class WordCountMapper extends Mapper<LongWritable
Text, Text, IntWritable> {
    private final static IntWritable one = new IntWritable(1);
    public void map(LongWritable key, Text value, Context context)
            throws IOException, InterruptedException{
        String[] words = value.toString().toLowerCase().
                split("\\W+");
```

```
        for(String word : words){
            if(!word.isEmpty()){
                context.write(new Text(word), one);
            }
        }
    }
}
```

Reducer:

```
public class WordCountReducer extends Reducer<Text, IntWritable,
        Text, IntWritable>{
    private Map<Text, Integer> countMap = new HashMap<>();

    public void reduce(Text key, Iterable<IntWritable> values,
                       Context context) throws
            IOException, InterruptedException{
        int sum = 0;
        for (IntWritable val : values){
            sum += val.get();
        }
        countMap.put(new Text(key), sum);
    }

    protected void cleanup(Context context) throws IOException,
            InterruptedException{
        List<Map.Entry<Text, Integer>> sorted = new ArrayList<>(
                .Map.entrySet());
        sorted.sort((e1, e2) -> e2.getValue().compareTo(e1.getValue()));

        int count = 0;
        for(Map.Entry<Text, Integer> entry : sorted){
            if(count++ == 10) break;
            context.write(entry.getKey(), new IntWritable(
```