LABORATORY PROGRAM – 7

For a given Text file, Create a Map Reduce program to sort the content in an alphabetic order listing only top 10 maximum occurrences of words.

OBSERVATION
CODE, COMMAND WITH OUTPUT

Driver Code (TopNDriver.java)

```
package samples.topn;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class TopNDriver {

   public static void main(String[] args) throws Exception {
      if (args.length != 3) {
         System.err.println("Usage: TopNDriver <in> <temp-out> <final-out>");
         System.exit(2);
      }

      Configuration conf = new Configuration();

      // === Job 1: Word Count ===
      Job wcJob = Job.getInstance(conf, "word count");
      wcJob.setJarByClass(TopNDriver.class);
      wcJob.setMapperClass(WordCountMapper.class);
      wcJob.setCombinerClass(WordCountReducer.class);
      wcJob.setReducerClass(WordCountReducer.class);
      wcJob.setOutputKeyClass(Text.class);
      wcJob.setOutputValueClass(IntWritable.class);

      FileInputFormat.addInputPath(wcJob, new Path(args[0]));
      Path tempDir = new Path(args[1]);
      FileOutputFormat.setOutputPath(wcJob, tempDir);

      if (!wcJob.waitForCompletion(true)) {
         System.exit(1);
      }

      // === Job 2: Top N ===
      Job topJob = Job.getInstance(conf, "top 10 words");
      topJob.setJarByClass(TopNDriver.class);
      topJob.setMapperClass(TopNMapper.class);
      topJob.setReducerClass(TopNReducer.class);
      topJob.setMapOutputKeyClass(IntWritable.class);
      topJob.setMapOutputValueClass(Text.class);
      topJob.setOutputKeyClass(Text.class);
      topJob.setOutputValueClass(IntWritable.class);

      FileInputFormat.addInputPath(topJob, tempDir);
      FileOutputFormat.setOutputPath(topJob, new Path(args[2]));

      System.exit(topJob.waitForCompletion(true) ? 0 : 1);
   }
```

```
}
```

Mapper Code (WordCountMapper.java)

```
package samples.topn;

import java.io.IOException;
import java.util.StringTokenizer;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

public class WordCountMapper
   extends Mapper<Object, Text, Text, IntWritable> {

   private final static IntWritable ONE = new IntWritable(1);
   private Text word = new Text();
   // characters to normalize into spaces
   private String tokens = "[_|$#<>\\^=\\[\\]\\*/\\\\,;.\\-:()?!\"']";

   @Override
   protected void map(Object key, Text value, Context context)
      throws IOException, InterruptedException {

      // clean & tokenize
      String clean = value.toString()
                     .toLowerCase()
                     .replaceAll(tokens, " ");
      StringTokenizer itr = new StringTokenizer(clean);
      while (itr.hasMoreTokens()) {
         word.set(itr.nextToken().trim());
         context.write(word, ONE);
      }
   }
}
```

Mapper Code (TopNMapper.java)

```
package samples.topn;

import java.io.IOException;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

public class TopNMapper
   extends Mapper<Object, Text, IntWritable, Text> {

   private IntWritable count = new IntWritable();
   private Text word = new Text();

   @Override
   protected void map(Object key, Text value, Context context)
      throws IOException, InterruptedException {

      // input line: word \t count
      String[] parts = value.toString().split("\\t");
      if (parts.length == 2) {
         word.set(parts[0]);
         count.set(Integer.parseInt(parts[1]));
         // emit count → word, so Hadoop sorts by count
         context.write(count, word);
      }
   }
}
```

Reducer Code (WordCountReducer.java)

```
package samples.topn;
```

```
import java.io.IOException;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

public class WordCountReducer
   extends Reducer<Text, IntWritable, Text, IntWritable> {

   @Override
   protected void reduce(Text key, Iterable<IntWritable> values, Context context)
      throws IOException, InterruptedException {

      int sum = 0;
      for (IntWritable val : values) {
         sum += val.get();
      }
      context.write(key, new IntWritable(sum));
   }
}
```

Reducer Code (TopNReducer.java)

```
package samples.topn;

import java.io.IOException;
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;
import java.util.Map;
import java.util.TreeMap;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

public class TopNReducer
   extends Reducer<IntWritable, Text, Text, IntWritable> {

   // TreeMap with descending order of keys (counts)
   private TreeMap<Integer, List<String>> countMap =
      new TreeMap<>(Collections.reverseOrder());

   @Override
   protected void reduce(IntWritable key, Iterable<Text> values, Context context)
      throws IOException, InterruptedException {

      int cnt = key.get();
      List<String> words = countMap.getOrDefault(cnt, new ArrayList<>());
      for (Text w : values) {
         words.add(w.toString());
      }
      countMap.put(cnt, words);
   }

   @Override
   protected void cleanup(Context context)
      throws IOException, InterruptedException {

      // collect top 10 word→count pairs
      List<WordCount> topList = new ArrayList<>();
      int seen = 0;
      for (Map.Entry<Integer, List<String>> entry : countMap.entrySet()) {
         int cnt = entry.getKey();
         for (String w : entry.getValue()) {
            topList.add(new WordCount(w, cnt));
            seen++;
            if (seen == 10) break;
         }
```

```
            if (seen == 10) break;
        }

        // sort these 10 entries alphabetically by word
        Collections.sort(topList, (a, b) -> a.word.compareTo(b.word));

        // emit final top 10 in alphabetical order
        for (WordCount wc : topList) {
            context.write(new Text(wc.word), new IntWritable(wc.count));
        }
    }

    // helper class
    private static class WordCount {
        String word;
        int count;
        WordCount(String w, int c) { word = w; count = c; }
    }
}
```

```
C:\hadoop-3.3.0\sbin>jps
11072 DataNode
20528 Jps
5620 ResourceManager
15532 NodeManager
6140 NameNode

C:\hadoop-3.3.0\sbin>hdfs dfs -mkdir /input_dir

C:\hadoop-3.3.0\sbin>hdfs dfs -ls /
Found 1 items
drwxr-xr-x   - Anusree supergroup          0 2021-05-08 19:46 /input_dir

C:\hadoop-3.3.0\sbin>hdfs dfs -copyFromLocal C:\input.txt /input_dir

C:\hadoop-3.3.0\sbin>hdfs dfs -ls /input_dir
Found 1 items
-rw-r--r--   1 Anusree supergroup         36 2021-05-08 19:48 /input_dir/input.txt

C:\hadoop-3.3.0\sbin>hdfs dfs -cat /input_dir/input.txt
hello
world
hello
hadoop
bye
```

```
C:\hadoop-3.3.0\sbin>hadoop jar C:\sort.jar samples.topn.TopN /input_dir/input.txt /output_dir
2021-05-08 19:54:54,582 INFO client.DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager at /0.0.0.0:8032
2021-05-08 19:54:55,291 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/Anusree/.staging/job_1620483374279_0001
2021-05-08 19:54:55,821 INFO input.FileInputFormat: Total input files to process : 1
2021-05-08 19:54:56,261 INFO mapreduce.JobSubmitter: number of splits:1
2021-05-08 19:54:56,552 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1620483374279_0001
2021-05-08 19:54:56,552 INFO mapreduce.JobSubmitter: Executing with tokens: []
2021-05-08 19:54:56,843 INFO conf.Configuration: resource-types.xml not found
2021-05-08 19:54:56,843 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
2021-05-08 19:54:57,387 INFO impl.YarnClientImpl: Submitted application application_1620483374279_0001
2021-05-08 19:54:57,507 INFO mapreduce.Job: The url to track the job: http://LAPTOP-JG329ESD:8088/proxy/application_1620483374279_0001/
2021-05-08 19:54:57,508 INFO mapreduce.Job: Running job: job_1620483374279_0001
2021-05-08 19:55:13,792 INFO mapreduce.Job: Job job_1620483374279_0001 running in uber mode : false
2021-05-08 19:55:13,794 INFO mapreduce.Job:  map 0% reduce 0%
2021-05-08 19:55:20,020 INFO mapreduce.Job:  map 100% reduce 0%
2021-05-08 19:55:27,116 INFO mapreduce.Job:  map 100% reduce 100%
2021-05-08 19:55:33,199 INFO mapreduce.Job: Job job_1620483374279_0001 completed successfully
2021-05-08 19:55:33,334 INFO mapreduce.Job: Counters: 54
        File System Counters
                FILE: Number of bytes read=65
                FILE: Number of bytes written=530397
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=142
                HDFS: Number of bytes written=31
                HDFS: Number of read operations=8
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
                HDFS: Number of bytes read erasure-coded=0
```

```
C:\hadoop-3.3.0\sbin>hdfs dfs -cat /output_dir/*
hello   2
hadoop  1
world   1
bye     1

C:\hadoop-3.3.0\sbin>
```