6. Create a Map Reduce program to
a) find average temperature for each year from NCDC data set.
b) find the mean max temperature for every month.

Dataset: **https://github.com/tomwhite/hadoop-book/tree/master/input/ncdc/all**

Driver code:

```
package averagetemp_amit;


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 AverageDriver {

  public static void main(String[] args) throws Exception {

    if (args.length != 2) {

      System.err.println("Please Enter the input and output parameters");

      System.exit(-1);

    }

    Job job = new Job();

    job.setJarByClass(AverageDriver.class);

    job.setJobName("Max temperature");

    FileInputFormat.addInputPath(job, new Path(args[0]));

    FileOutputFormat.setOutputPath(job, new Path(args[1]));

    job.setMapperClass(AverageMapper.class);

    job.setReducerClass(AverageReducer.class);

    job.setOutputKeyClass(Text.class);

    job.setOutputValueClass(IntWritable.class);
    System.exit(job.waitForCompletion(true) ? 0 : 1);
  }
}
```

Mapper:

```
package averagetemp_amit;

import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

public class AverageMapper extends Mapper<LongWritable, Text, Text, IntWritable> {
  public static final int MISSING = 9999;

  public void map(LongWritable key, Text value, Mapper<LongWritable, Text, Text, 
IntWritable>.Context context) throws IOException, InterruptedException {
    int temperature;
    String line = value.toString();
    String year = line.substring(15, 19);
    if (line.charAt(87) == '+') {
      temperature = Integer.parseInt(line.substring(88, 92));
    } else {
      temperature = Integer.parseInt(line.substring(87, 92));
    }
    String quality = line.substring(92, 93);
    if (temperature != 9999 && quality.matches("[01459]"))
      context.write(new Text(year), new IntWritable(temperature));
  }
}
```

Reducer:

```
package averagetemp_amit;


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


public class AverageReducer extends Reducer<Text, IntWritable, Text, IntWritable> {
  public void reduce(Text key, Iterable<IntWritable> values, Reducer<Text, IntWritable, Text, 
IntWritable>.Context context) throws IOException, InterruptedException {
    int max_temp = 0;
    int count = 0;
    for (IntWritable value : values) {
      max_temp += value.get();
      count++;
    }
    context.write(key, new IntWritable(max_temp / count));
  }
}
```

7. Create a Map Reduce program to sort the content in an alphabetic order listing only top 10 maximum occurrences of words.

Driver:

```
import java.io.IOException;
import java.util.StringTokenizer;
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.Mapper;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.util.GenericOptionsParser;
public class TopN {
        public static void main(String[] args) throws Exception {
         Configuration conf = new Configuration();
         String[] otherArgs = (new GenericOptionsParser(conf, args)).getRemainingArgs();
         if (otherArgs.length != 2) {
           System.err.println("Usage: TopN <in> <out>");
           System.exit(2);
         }
         Job job = Job.getInstance(conf);
         job.setJobName("Top N");
         job.setJarByClass(TopN.class);
         job.setMapperClass(TopNMapper.class);
         job.setReducerClass(TopNReducer.class);
         job.setOutputKeyClass(Text.class);
```

```
            job.setOutputValueClass(IntWritable.class);
            FileInputFormat.addInputPath(job, new Path(otherArgs[0]));
            FileOutputFormat.setOutputPath(job, new Path(otherArgs[1]));
            System.exit(job.waitForCompletion(true) ? 0 : 1);
          }

          public static class TopNMapper extends Mapper<Object, Text, Text, IntWritable> {
            private static final IntWritable one = new IntWritable(1);
            private Text word = new Text();
            private String tokens = "[_|$#<>\\^=\\[\\]\\*/\\\\,;,.\\-:()?!\"']";
            public void map(Object key, Text value, Mapper<Object, Text, Text,
IntWritable>.Context context) throws IOException, InterruptedException {
              String cleanLine = value.toString().toLowerCase().replaceAll(this.tokens, " ");
              StringTokenizer itr = new StringTokenizer(cleanLine);
              while (itr.hasMoreTokens()) {
                this.word.set(itr.nextToken().trim());
                context.write(this.word, one);
              }
            }
          }
        }
```

Mapper:

```
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 TopNMapper extends Mapper<Object, Text, Text, IntWritable> {
  private static final IntWritable one = new IntWritable(1);
  private Text word = new Text();
  private String tokens = "[_|$#<>\\^=\\[\\]\\*/\\\\,;,.\\-:()?!\"']";

  public void map(Object key, Text value, Mapper<Object, Text, Text, IntWritable>.Context
context) throws IOException, InterruptedException {
    String cleanLine = value.toString().toLowerCase().replaceAll(this.tokens, " ");
    StringTokenizer itr = new StringTokenizer(cleanLine);
    while (itr.hasMoreTokens()) {
      this.word.set(itr.nextToken().trim());
      context.write(this.word, one);
    }
  }
}
```

Combiner:

```
import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;
public class TopNCombiner extends Reducer<Text, IntWritable, Text, IntWritable> {
  public void reduce(Text key, Iterable<IntWritable> values, Reducer<Text, IntWritable, Text,
IntWritable>.Context context) throws IOException, InterruptedException {
    int sum = 0;
    for (IntWritable val : values)
      sum += val.get();
    context.write(key, new IntWritable(sum));
  }
}
```

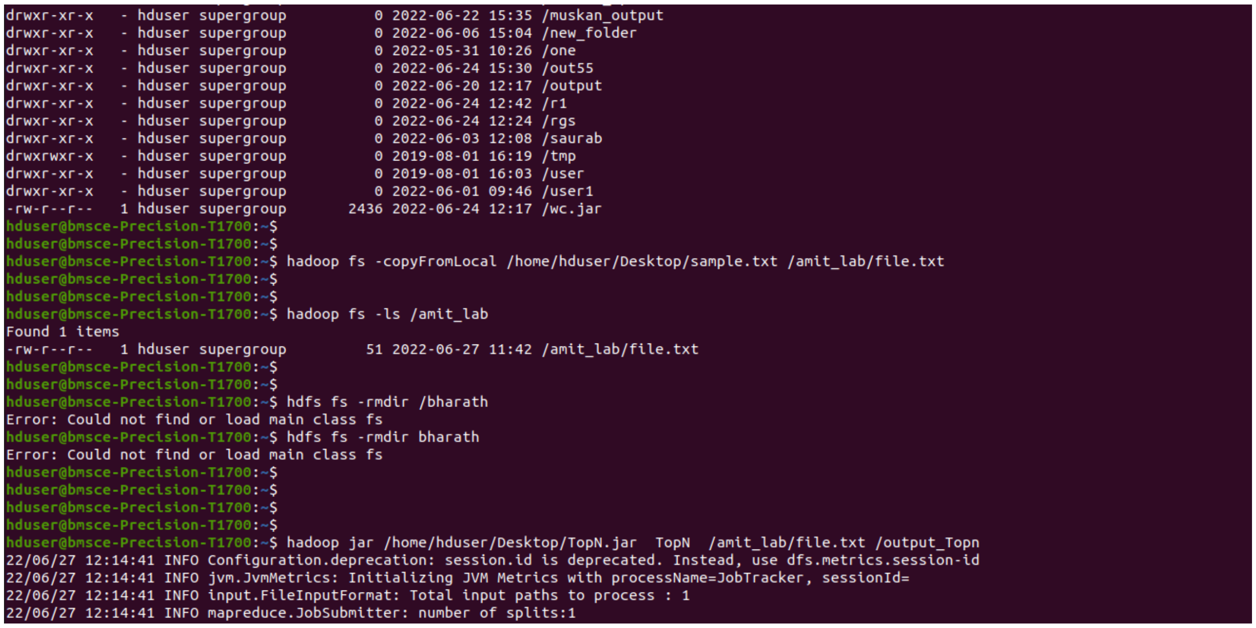
Reducer:

```
import java.io.IOException;
import java.util.HashMap;
import java.util.Map;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;
import utils.MiscUtils;


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


  public void reduce(Text key, Iterable<IntWritable> values, Reducer<Text, IntWritable, Text,
IntWritable>.Context context) throws IOException, InterruptedException {
    int sum = 0;
    for (IntWritable val : values)
      sum += val.get();
    this.countMap.put(new Text(key), new IntWritable(sum));
  }
  protected void cleanup(Reducer<Text, IntWritable, Text, IntWritable>.Context context) throws
IOException, InterruptedException {
    Map<Text, IntWritable> sortedMap = MiscUtils.sortByValues(this.countMap);
    int counter = 0;
    for (Text key : sortedMap.keySet()) {
      if (counter++ == 20)
        break;
      context.write(key, sortedMap.get(key));
    }
  }
}
```

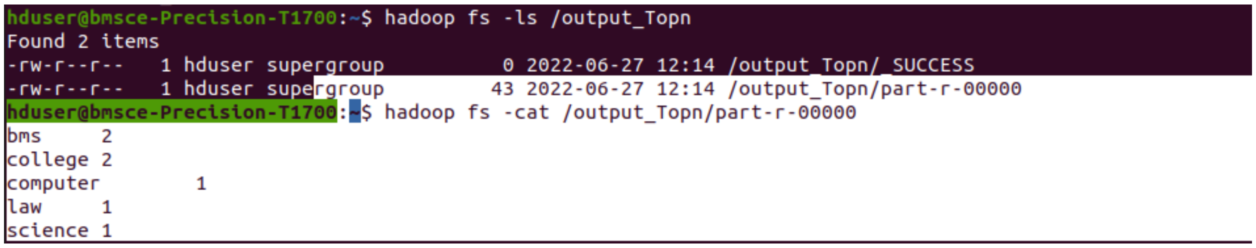
MiscUtils.java

```
package utils;
import java.util.*;
public class MiscUtils {
public static <K extends Comparable, V extends Comparable> Map<K, V> sortByValues(Map<K,
V> map) {
List<Map.Entry<K, V>> entries = new LinkedList<Map.Entry<K, V>>(map.entrySet());
Collections.sort(entries, new Comparator<Map.Entry<K, V>>() {
@Override
public int compare(Map.Entry<K, V> o1, Map.Entry<K, V> o2) {
return o2.getValue().compareTo(o1.getValue());
}
});
//LinkedHashMap will keep the keys in the order they are inserted

//which is currently sorted on natural ordering

Map<K, V> sortedMap = new LinkedHashMap<K, V>();
for (Map.Entry<K, V> entry : entries) {
sortedMap.put(entry.getKey(), entry.getValue());
}
return sortedMap;
}
}
```

Output:



```
drwxr-xr-x   - hduser supergroup          0 2022-06-22 15:35 /muskan_output
drwxr-xr-x   - hduser supergroup          0 2022-06-06 15:04 /new_folder
drwxr-xr-x   - hduser supergroup          0 2022-05-31 10:26 /one
drwxr-xr-x   - hduser supergroup          0 2022-06-24 15:30 /out55
drwxr-xr-x   - hduser supergroup          0 2022-06-20 12:17 /output
drwxr-xr-x   - hduser supergroup          0 2022-06-24 12:42 /r1
drwxr-xr-x   - hduser supergroup          0 2022-06-24 12:24 /rgs
drwxr-xr-x   - hduser supergroup          0 2022-06-03 12:08 /saurab
drwxrwxr-x   - hduser supergroup          0 2019-08-01 16:19 /tmp
drwxr-xr-x   - hduser supergroup          0 2019-08-01 16:03 /user
drwxr-xr-x   - hduser supergroup          0 2022-06-01 09:46 /user1
-rw-r--r--   1 hduser supergroup       2436 2022-06-24 12:17 /wc.jar
hduser@bmsce-Precision-T1700:~$
hduser@bmsce-Precision-T1700:~$
hduser@bmsce-Precision-T1700:~$ hadoop fs -copyFromLocal /home/hduser/Desktop/sample.txt /amit_lab/file.txt
hduser@bmsce-Precision-T1700:~$
hduser@bmsce-Precision-T1700:~$
hduser@bmsce-Precision-T1700:~$ hadoop fs -ls /amit_lab
Found 1 items
-rw-r--r--   1 hduser supergroup         51 2022-06-27 11:42 /amit_lab/file.txt
hduser@bmsce-Precision-T1700:~$
hduser@bmsce-Precision-T1700:~$
hduser@bmsce-Precision-T1700:~$ hdfs fs -rmdir /bharath
Error: Could not find or load main class fs
hduser@bmsce-Precision-T1700:~$ hdfs fs -rmdir bharath
Error: Could not find or load main class fs
hduser@bmsce-Precision-T1700:~$
hduser@bmsce-Precision-T1700:~$
hduser@bmsce-Precision-T1700:~$
hduser@bmsce-Precision-T1700:~$
hduser@bmsce-Precision-T1700:~$ hadoop jar /home/hduser/Desktop/TopN.jar  TopN  /amit_lab/file.txt /output_Topn
22/06/27 12:14:41 INFO Configuration.deprecation: session.id is deprecated. Instead, use dfs.metrics.session-id
22/06/27 12:14:41 INFO jvm.JvmMetrics: Initializing JVM Metrics with processName=JobTracker, sessionId=
22/06/27 12:14:41 INFO input.FileInputFormat: Total input paths to process : 1
22/06/27 12:14:41 INFO mapreduce.JobSubmitter: number of splits:1
```



```
hduser@bmsce-Precision-T1700:~$ hadoop fs -ls /output_Topn
Found 2 items
-rw-r--r--   1 hduser supergroup          0 2022-06-27 12:14 /output_Topn/_SUCCESS
-rw-r--r--   1 hduser supergroup         43 2022-06-27 12:14 /output_Topn/part-r-00000
hduser@bmsce-Precision-T1700:~$ hadoop fs -cat /output_Topn/part-r-00000
bms     2
college 2
computer        1
law     1
science 1
```

8. Create a Map Reduce program to demonstrating join operation.

DeptEmpStrength.txt

Dept_ID	Total_Employee
A11	50
B12	100
C13	250

DeptName.txt

Dept_ID	Dept_Name
A11	Finance
B12	HR
C13	Manufacturing

Driver:

```
package MapReduceJoin;

import org.apache.hadoop.conf.Configured;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.*;
import org.apache.hadoop.mapred.lib.MultipleInputs;
import org.apache.hadoop.util.*;


public class JoinDriver extends Configured implements Tool {
	public static class KeyPartitioner implements Partitioner<TextPair, Text> {
		@Override
		public void configure(JobConf job) {}

		@Override
		public int getPartition(TextPair key, Text value, int numPartitions) {
			return (key.getFirst().hashCode() & Integer.MAX_VALUE) %
numPartitions;
		}
	}

	@Override
	public int run(String[] args) throws Exception {

		if (args.length != 3) {
			System.out.println("Usage: <Department Emp Strength input>
<Department Name input> <output>");
			return -1;
		}

		JobConf conf = new JobConf(getConf(), getClass());
		conf.setJobName("Join 'Department Emp Strength input' with 'Department
Name input'");

		Path AInputPath = new Path(args[0]);
		Path BInputPath = new Path(args[1]);
		Path outputPath = new Path(args[2]);

		MultipleInputs.addInputPath(conf, AInputPath, TextInputFormat.class,
DeptNameMapper.class);
```

```
                MultipleInputs.addInputPath(conf, BInputPath, TextInputFormat.class,
DeptEmpStrengthMapper.class);

                FileOutputFormat.setOutputPath(conf, outputPath);

                conf.setPartitionerClass(KeyPartitioner.class);
                conf.setOutputValueGroupingComparator(TextPair.FirstComparator.class);

                conf.setMapOutputKeyClass(TextPair.class);

                conf.setReducerClass(JoinReducer.class);

                conf.setOutputKeyClass(Text.class);

                JobClient.runJob(conf);

                return 0;
        }

        public static void main(String[] args) throws Exception {

                int exitCode = ToolRunner.run(new JoinDriver(), args);
                System.exit(exitCode);
        }
}
```

Mapper:

DeptEmpStrengthMapper.java

```
package MapReduceJoin;


import java.io.IOException;

import java.util.Iterator;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.FSDataInputStream;

import org.apache.hadoop.fs.FSDataOutputStream;

import org.apache.hadoop.fs.FileSystem;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapred.*;


import org.apache.hadoop.io.IntWritable;


public class DeptEmpStrengthMapper extends MapReduceBase implements
Mapper<LongWritable, Text, TextPair, Text> {


	@Override

	public void map(LongWritable key, Text value, OutputCollector<TextPair, Text> output,
Reporter reporter)

				throws IOException

	{


		String valueString = value.toString();

		String[] SingleNodeData = valueString.split("\t");
```

```
                    output.collect(new TextPair(SingleNodeData[0], "1"), new
Text(SingleNodeData[1]));
          }
}
```

DeptNameMapper.java

```
package MapReduceJoin;

import java.io.IOException;

import org.apache.hadoop.io.*;
import org.apache.hadoop.mapred.*;

public class DeptNameMapper extends MapReduceBase implements Mapper<LongWritable,
Text, TextPair, Text>  {

          @Override
          public void map(LongWritable key, Text value, OutputCollector<TextPair, Text> output,
Reporter reporter)
                              throws IOException
          {
                    String valueString = value.toString();
                    String[] SingleNodeData = valueString.split("\t");
                    output.collect(new TextPair(SingleNodeData[0], "0"), new
Text(SingleNodeData[1]));
          }
}
```

Reducer:

```
package MapReduceJoin;


import java.io.IOException;
import java.util.Iterator;


import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.*;


public class JoinReducer extends MapReduceBase implements Reducer<TextPair, Text, Text,
Text> {


	@Override
	public void reduce (TextPair key, Iterator<Text> values, OutputCollector<Text, Text>
output, Reporter reporter)
			throws IOException
	{


		Text nodeId = new Text(values.next());
		while (values.hasNext()) {
			Text node = values.next();
			Text outValue = new Text(nodeId.toString() + "\t\t" + node.toString());
			output.collect(key.getFirst(), outValue);
		}
	}
}
```

Jar link:

https://github.com/amitkumar70512/BDA_LAB/blob/main/Lab8/MapReduceJoin/MapReduceJoin.jar