Big Data Assignment 1

Code:

Mapper Code-

```
// Importing libraries
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.mapred.MapReduceBase;
import org.apache.hadoop.mapred.Mapper;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reporter;

public class WCMapper extends MapReduceBase implements Mapper<LongWritable,

        Text, Text, IntWritable> {

        // Map function
        public void map(LongWritable key, Text value, OutputCollector<Text,
                        IntWritable> output, Reporter rep) throws IOException
        {

                String line = value.toString();

                // Splitting the line on spaces
                for (String word : line.split(" "))
                {
                        if (word.length() > 0)
                        {
```

```
                        output.collect(new Text(word), new IntWritable(1));
                    }
                }
            }
}
```

Reducer Code –

```
// Importing libraries
import java.io.IOException;
import java.util.Iterator;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reducer;
import org.apache.hadoop.mapred.Reporter;

public class WCReducer extends MapReduceBase implements Reducer<Text,
                                                    IntWritable, Text,
IntWritable> {

    // Reduce function
    public void reduce(Text key, Iterator<IntWritable> value,
                        OutputCollector<Text, IntWritable> output,
                                    Reporter rep) throws IOException
    {

        int count = 0;

        // Counting the frequency of each words
```

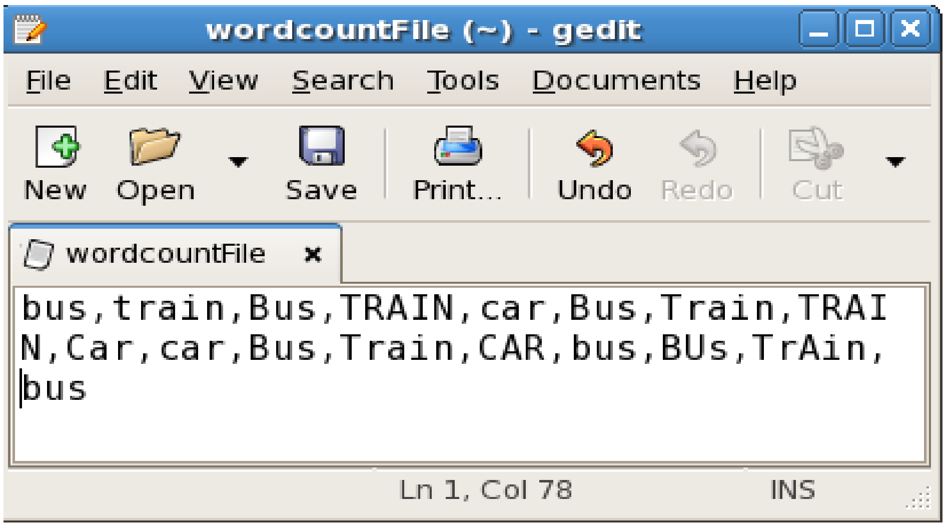
```
		while (value.hasNext())
		{
			IntWritable i = value.next();
			count += i.get();
		}

		output.collect(key, new IntWritable(count));
	}
}
```

Driver Code –

```
// Importing libraries
import java.io.IOException;
import org.apache.hadoop.conf.Configured;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.FileInputFormat;
import org.apache.hadoop.mapred.FileOutputFormat;
import org.apache.hadoop.mapred.JobClient;
import org.apache.hadoop.mapred.JobConf;
import org.apache.hadoop.util.Tool;
import org.apache.hadoop.util.ToolRunner;

public class WCDriver extends Configured implements Tool {

	public int run(String args[]) throws IOException
	{
		if (args.length < 2)
		{
```

```
                System.out.println("Please give valid inputs");
                return -1;
            }

            JobConf conf = new JobConf(WCDriver.class);
            FileInputFormat.setInputPaths(conf, new Path(args[0]));
            FileOutputFormat.setOutputPath(conf, new Path(args[1]));
            conf.setMapperClass(WCMapper.class);
            conf.setReducerClass(WCReducer.class);
            conf.setMapOutputKeyClass(Text.class);
            conf.setMapOutputValueClass(IntWritable.class);
            conf.setOutputKeyClass(Text.class);
            conf.setOutputValueClass(IntWritable.class);
            JobClient.runJob(conf);
            return 0;
        }

        // Main Method
        public static void main(String args[]) throws Exception
        {
            int exitCode = ToolRunner.run(new WCDriver(), args);
            System.out.println(exitCode);
        }
}
```

Output:


wordcountFile (~) - gedit
File Edit View Search Tools Documents Help
New Open Save Print... Undo Redo Cut
wordcountFile
Ln 1, Col 78 INS

```
bus,train,Bus,TRAIN,car,Bus,Train,TRAI
N,Car,car,Bus,Train,CAR,bus,BUs,TrAin,
bus
```

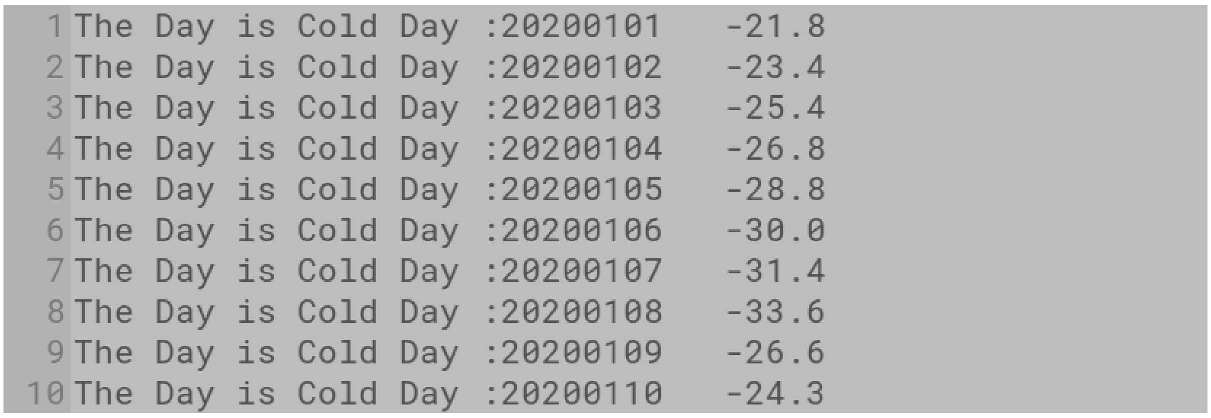
Big Data Assignment 3

Code:

```
import java.io.IOException;
import java.util.Iterator;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.conf.Configuration;

public class MyMaxMin {

        public static class MaxTemperatureMapper extends
                        Mapper<LongWritable, Text, Text, Text> {



public static final int MISSING = 9999;

        @Override
                public void map(LongWritable arg0, Text Value, Context context)
                                throws IOException, InterruptedException {
```

```
        String line = Value.toString();

            // Check for the empty line
            if (!(line.length() == 0)) {

                String date = line.substring(6, 14);



                float temp_Max = Float.parseFloat(line.substring(39,
45).trim());



                float temp_Min = Float.parseFloat(line.substring(47,
53).trim());

                if (temp_Max > 30.0) {

                    // Hot day
                    context.write(new Text("The Day is Hot Day :" +
date),
                                                    new
Text(String.valueOf(temp_Max)));
                }

                if (temp_Min < 15) {

                    // Cold day
                    context.write(new Text("The Day is Cold Day :" +
date),
                                                            new
Text(String.valueOf(temp_Min)));
                                }
                        }
                }

        }

        public static class MaxTemperatureReducer extends
                        Reducer<Text, Text, Text, Text> {

                public void reduce(Text Key, Iterator<Text> Values, Context context)
                                throws IOException, InterruptedException {

                        String temperature = Values.next().toString();
                        context.write(Key, new Text(temperature));
                }
        }

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

                Configuration conf = new Configuration();

                Job job = new Job(conf, "weather example");

                job.setJarByClass(MyMaxMin.class);

                job.setMapOutputKeyClass(Text.class);

                job.setMapOutputValueClass(Text.class);
                job.setMapperClass(MaxTemperatureMapper.class);
                // Defining the reducer class name
                job.setReducerClass(MaxTemperatureReducer.class);
                job.setInputFormatClass(TextInputFormat.class);

                job.setOutputFormatClass(TextOutputFormat.class);

                Path OutputPath = new Path(args[1]);

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

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

                OutputPath.getFileSystem(conf).delete(OutputPath);

                System.exit(job.waitForCompletion(true) ? 0 : 1);

        }
}
```

Output:



```
The Day is Cold Day :20200101   -21.8
The Day is Cold Day :20200102   -23.4
The Day is Cold Day :20200103   -25.4
The Day is Cold Day :20200104   -26.8
The Day is Cold Day :20200105   -28.8
The Day is Cold Day :20200106   -30.0
The Day is Cold Day :20200107   -31.4
The Day is Cold Day :20200108   -33.6
The Day is Cold Day :20200109   -26.6
The Day is Cold Day :20200110   -24.3
```