Name of the student:	Tanmay Prashant Rane	**Roll No.**	8031
Practical Number:	3	**Date of Practical:**	
Relevant CO's	At the end of the course students will be able to use tools like hadoop and NoSQL to solve big data related problems.		
Sign here to indicate that you have read all the relevant material provided before attempting this practical			Sign:

Practical grading using Rubrics

Indicator	Very Poor	Poor	Average	Good	Excellent
Timeline (2)	More than a session late (0)	NA	NA	NA	Early or on time (2)
Code design (2)	N/A	Very poor code design with no comments and indentation(0.5)	Poor code design with very comments and indentation (1)	Design with good coding standards (1.5)	Accurate design with better coding satndards (2)
Performance (4)	Unable to perform the experiment (0)	Able to partially perform the experiment (1)	Able to perform the experiment for certain use cases (2)	Able to perform the experiment considering most of the use cases (3)	Able to perform the experiment considering all use cases (4)
Postlab (2)	No Execution(0)	N/A	Partially Executed (1)	N/A	Fully Executed (2)

Total Marks (10)	Sign of instructor

Practical

Course title: Big Data Analytics
Course term: 2018-2019
Instructor name: Saurabh Kulkarni

Problem Statement: Find year-wise maximum temperature from the given dataset using map-reduce.

Theory:Explain the working of finding maximum temperature using map reduce with small example and diagrams

The Map Function: The key is the offset of the beginning of the line from the beginning of the file. Map function setting up the data in such a way that the reduce function can do its work on it. The map function is also a good place to drop bad records.

So, generally, we filter out the necessary data that we need to process. To provide the body of Map Function we need to extend Mapper class. To understand Map Function better, let's look at an example where we are considering **NCDC raw data**:

We need to find the maximum temperature for each year.

```
0067011990999991950051507004…9999999N9+0000+99999999999…

0043011990999991950051512004…9999999N9+0022+199999999999…

0043011990999991950051518004…9999999N9-0011+99999999999…

0043012650999991949032412004…0500001N9+0111+99999999999…

0043012650999991949032418004…0500001N9+0078+99999999999…
```

So, the input for out map function is something like this.

```
(0, 0067011990999991950051507004…9999999N9+000099999999999…)

(106, 0043011990999991950051512004…9999999N9+0022+99999999999…)

(212, 0043011990999991950051518004…9999999N9-0011+99999999999…)

(318, 0043012650999991949032412004…0500001N9+0111+99999999999…)

(424, 0043012650999991949032418004…0500001N9+0078+99999999999…)
```

The keys are the line offsets within the file, which we ignore in our map function. The

map function merely extracts the year and the air temperature (indicated in bold text), and emits them as its output (the temperature values have been interpreted as integers):

(1950, 0)

(1950, 22)

(1950, −11)

(1949, 111)

(1949, 78)

The Mapper class is a generic type, with four formal type parameters that specify the input key, input value, output key, and output value types of the map function.
For the present example, the input key is a long integer offset, the input value is a line of text, the output key is a year, and the output value is an air temperature (an integer). The map() method also provides an instance of Context to write the output to. In this case, we write the year as a Text object.

The Reduce Function: In Reduce Function we perform actual computation. The operation that we need to perform totally depends on the user's user case. The Input for the Reduce function is like this (Key, List(values)).
Example Continue: Here, input for the Reduce Function is something like this:

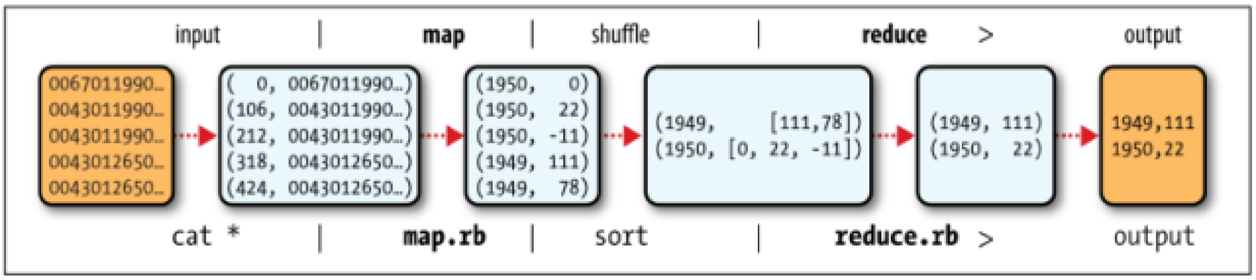
(1949, [111, 78])

(1950, [0, 22, −11])

gain, four formal type parameters are used to specify the input and output types, this time for the reduce function. **The input types of the reduce function must match the output types of the map function**.
The output of our program is something like this:

(1949, 111)

(1950, 22)



Code:

code for mapper:

Code for Reducer:

Code for Driver Class:

```
//MAPPER
import java.io.IOException;

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

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

        public void map(LongWritable key, Text value, Context context)
                        throws IOException, InterruptedException
        {
                String line = value.toString();
                String year = line.substring(15,19);
                int temperature;
                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 != MISSING && quality.matches("[01459]"))
                        context.write(new Text(year),new IntWritable(temperature));
        }

}

//REDUCER
import java.io.IOException;


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


public class HighestReducer extends Reducer<Text, IntWritable, Text, IntWritable> {
```

```
public void reduce(Text key, Iterable<IntWritable> values, Context context)
                        throws IOException, InterruptedException {
                // process values
                int max_temp = 0;
                for(IntWritable val:values) {
                        max_temp = Math.max(max_temp,val.get());
                }

                context.write(key, new IntWritable(max_temp));
        }

}

//DRIVER

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.io.IntWritable;
import java.util.*;

public class MaxTemperature {

        public static void main(String[] args) throws Exception {
                Configuration conf = new Configuration();
                Job job = Job.getInstance(conf, "maxtemp");
                job.setJarByClass(MaxTemperature.class);
                // TODO: specify a mapper
                job.setMapperClass(HighestMapper.class);
                // TODO: specify a reducer
                job.setReducerClass(HighestReducer.class);

                // TODO: specify output types
                job.setOutputKeyClass(Text.class);
                job.setOutputValueClass(IntWritable.class);

                // TODO: specify input and output DIRECTORIES (not files)
                FileInputFormat.setInputPaths(job, new
Path("/media/tanmay/Data/SEM-8/BDA/EXP3/Temperature_data"));
                FileOutputFormat.setOutputPath(job, new Path("/home/tanmay/out4"));

                if (!job.waitForCompletion(true)) {
                        System.out.println("Job Completed");
                        return;
                }
        }

}
```

PostLab:

Can we use combiner function in the above code? If yes, write the above code with combiner function.

Code for postlab question

```
//POSTLAB COMBINER
job.setCombinerClass(HighestReducer.class);
```