Course title: Big Data Analytics

Course term: 2016-2017 Practical

Name of the student:		**Roll No.**	
Practical Number:		**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

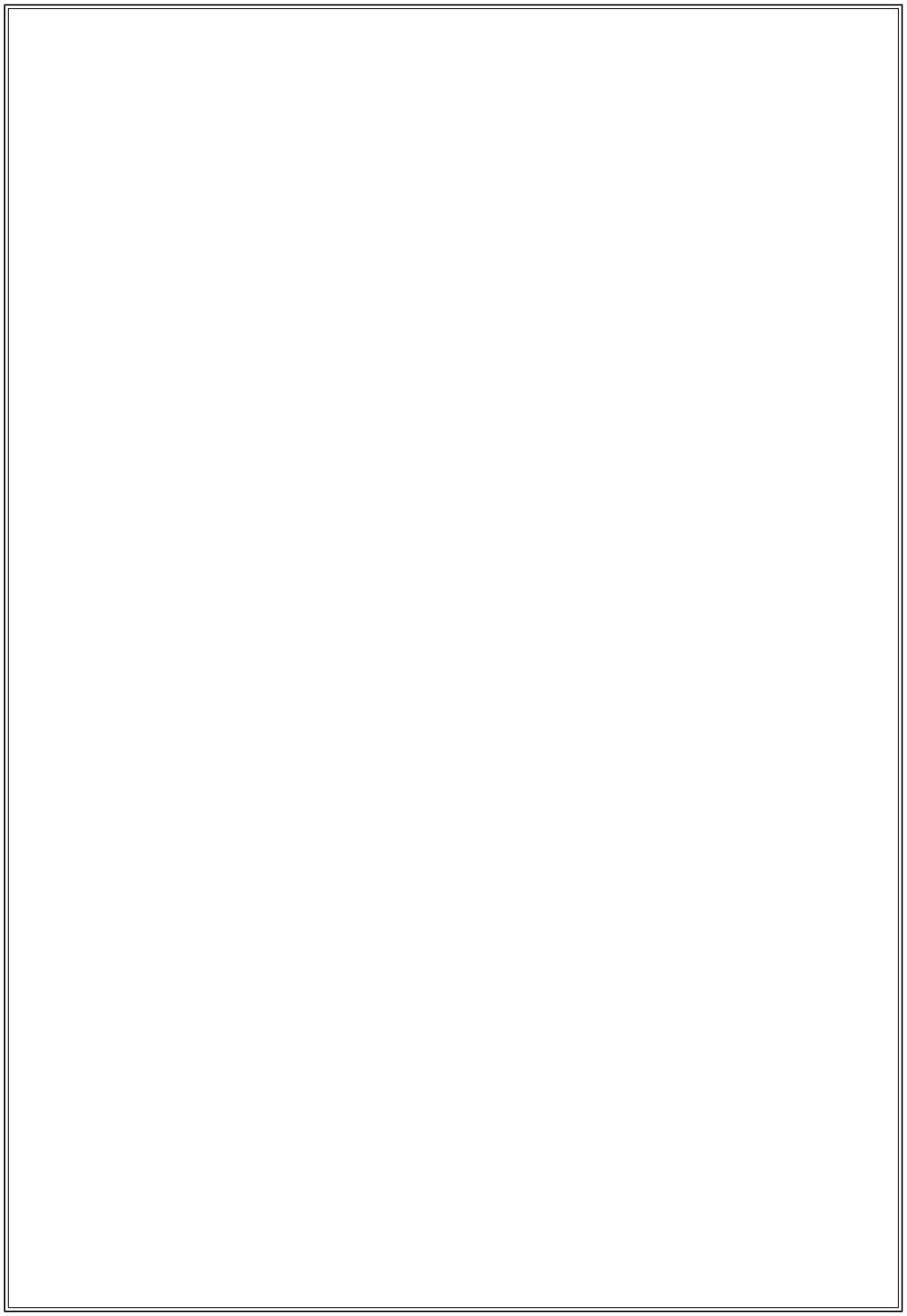
Indicator	**Very Poor**	**Poor**	**Average**	**Good**	**Excellent**
Timeline (2)	Practical not submitted (0)	More than two session late (0.5)	Two sessions late (1)	One session late (1.5)	Early or on time (2)
Code design (3)	N/A	Very poor code design(0)	poor design (1)	design with good coding standards (2)	Accurate Design with better coding standards(3)
Execution (3)	N/A	Very less execution (0)	little execution.(1)	Major execution(2)	Entire code execution (3)
Postlab (2)	No Execution(0)	N/A	Partially Executed (1)	N/A	Fully Executed (2)

Total Marks (10)	**Sign of instructor**

2020/04/16

Problem Statement: Counting number of words in given text file using map reduce.

Theory:Explain the working of word count using map reduce with small example and diagrams



Code:

code for mapper:

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

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

public class WCMapper extends Mapper<LongWritable, Text,
    red↪ Text, IntWritable> {
// Create object of type Text to hold strings created
    red↪ per word of given document
private Text word = new Text();

// Create final static variable of type IntWritable with
    red↪ value equal to 1 as according to algorithm map
    red↪ function puts 1 for each word encountered.
private final static IntWritable one = new IntWritable
    red↪ (1);

//write a map function here
public void map(LongWritable ikey, Text ivalue, Context
    red↪ context)
throws IOException, InterruptedException {
//Define a String type variable and assign value which
    red↪ is equal to string equivalent of Text value
    red↪ passed to map function
String line = ivalue.toString();
//Convert this variable to tokens using StringTokenizer
    red↪ class as it separates out each word of the
    red↪ document
StringTokenizer tokenizer = new StringTokenizer(line);
//Until there are tokens in StringTokenizer,
while(tokenizer.hasMoreTokens())
{
// set the Text object created in first line of the code
```

```
    red↪  in WCMapper to next token in the
    red↪ StringTokenizer
word.set(tokenizer.nextToken());
// Write this Text Variable and final static IntWritable
    red↪  Variable created to the context so that key-
    red↪ value pairs are generated.
context.write(word, one);
}
}
}
```

Code for Reducer:

Code for Driver Class:

PostLab:Find inverted index

In this assignment you have to implement a simple map reduce job that builds an inverted index on the set of input documents. An inverted index maps each word to a list of documents that contain the word, and additionally records the position of each occurrence of the word within the document. For the purpose of this assignment, the position will be based on counting words, not characters.

Ex: Assume below are the input Documents.
file1="data is good."
file2="data is not good?"

Output:
data (file1,1)(file2,1)
good (file1,3)(file2,4)
is (file1,2)(file2,2)
not (file2,3)
For more details on inverted indices, you can check out the Wikipedia page on inverted indices.
Now in this assignment you need to implement above map-reduce job.
Input: A set of documents

Output:
Map: word1 (filename, position)
word2 (filename, position)
word1 (filename, position)
and so on for each occurrence of each word.
Reduce: word1 (filename, position)(filename,position)
word2 (filename, position)
and so on for each word.
Code for getting file name in Hadoop, which can be used in the Map function:

```
String filename=null;
filename = ((FileSplit) context.
    red↪ getInputSplit()).getPath().
    red↪ getName();
```

Code for postlab question