**ASSIGNMENT : HADOOP**

**\*\*\*Step 1:🡪To Start Hadoop ,Open TERMINAL\*\*\***

i)cd hadoop

ii) start-dfs.sh-🡪start datanode and namenode

iii)start-yarn.sh🡪start resource manager

iv)jps🡪check 6 nodes are started or not

**\*\*\*step 2:->WordCount file Setup for Code**

1)Create Folder on Desktop and open that folder in eclipse.

2)File🡪New🡪Java Project🡪Give project name such as **WordCount** and check RE ,it should be JavaSE-11🡪also make changes in configure JRE .

3)To change configure JRE 🡪click on compiler🡪change level to 11🡪then click on apply and close.

4)After that click on next 🡪click Libraries

5)classPATH🡪Add External JRE-🡪Hadoop🡪Share🡪Hadoop🡪Common🡪select all JAR files and open .

6)Again click on classPATH🡪Add External JRE-🡪Hadoop🡪Share🡪Hadoop🡪MapReduce🡪select all JAR files and open .

7) click on finish

8)Now in project i.e WordCount🡪Go in Src and delete Modulo\_info.java file

9)Right click on WordCount Project🡪New🡪Class🡪Give name:WordCount🡪click on finish and then write your code here.

10)Again Right click on WordCount Project 🡪Export 🡪Java 🡪JAR file🡪click on next🡪click browse and go in folder and save that jar file name:- abc.jar in that folder .

11)click Next🡪Again click Next🡪Main Class🡪click on Browse🡪click on OK🡪Then finish.

**\*\*\* create input.txt file \*\*\***

1)Give input as per require and save as input.txt in folder

**\*\*\*To Run Code\*\*\***

1)Open TERMINAL in that folder

2)hadoop fs -mkdir /31445🡪folder 31445 is created on localhost:9870

3)hadoop fs -mkdir /31445/input 🡪created input folder in 31445 folder on local host

4)To see our folders are created or not open localhost:9870 🡪Click on utilities🡪click on Browse file System ,refresh ,and check.

5)hadoop fs -put input.txt /31445/input 🡪To put our input file on local host

6)hadoop jar abc.jar /31445/input /31445/output🡪Our output file is created on local host🡪to check our output🡪 clicl on part-r-000000 🡪click on head the file

7)hadoop dfs -cat /31445/output/\* 🡪to see output on terminal

**WordCount File**

**Code:🡪**

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.Reducer;

import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;

import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class WordCount {

public static class TokenizerMapper

extends Mapper<Object, Text, Text, IntWritable>{

private final static IntWritable one = new IntWritable(1);

private Text word = new Text();

public void map(Object key, Text value, Context context

) throws IOException, InterruptedException {

StringTokenizer itr = new StringTokenizer(value.toString());

while (itr.hasMoreTokens()) {

word.set(itr.nextToken());

context.write(word, one);

}

}

}

public static class IntSumReducer

extends Reducer<Text,IntWritable,Text,IntWritable> {

private IntWritable result = new IntWritable();

public void reduce(Text key, Iterable<IntWritable> values,

Context context

) throws IOException, InterruptedException {

int sum = 0;

for (IntWritable val : values) {

sum += val.get();

}

result.set(sum);

context.write(key, result);

}

}

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

Configuration conf = new Configuration();

Job job = Job.getInstance(conf, "word count");

job.setJarByClass(WordCount.class);

job.setMapperClass(TokenizerMapper.class);

job.setCombinerClass(IntSumReducer.class);

job.setReducerClass(IntSumReducer.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(IntWritable.class);

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

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

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

}

}

**Input.txt file🡪**

Varsha

Varsha

Poonam

Afrin

Poonam

**ouput🡪**

Varsha 2

Poonam 2

Afrin 1

**ASSIGNMENT : HADOOP**

**Process🡪same as Assignment 11**

**Weathermap file:🡪**

import java.io.IOException;  
  
import org.apache.hadoop.conf.Configuration;  
import org.apache.hadoop.fs.Path;  
import org.apache.hadoop.io.FloatWritable;  
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.TextInputFormat;  
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;  
  
public class weathermap {  
  
    public static class TokenizerMapper extends Mapper<Object, Text, Text, FloatWritable> {  
  
        private Text category = new Text();  
        private FloatWritable temperature = new FloatWritable();  
        private FloatWritable windSpeed = new FloatWritable();  
        private FloatWritable dewPoint = new FloatWritable();  
  
        public void map(Object key, Text value, Context context) throws IOException, InterruptedException {  
            String[] tokens = value.toString().split(" ");  
            float temp = Float.parseFloat(tokens[0]);  
            float wind = Float.parseFloat(tokens[1]);  
            float dew = Float.parseFloat(tokens[2]);  
  
            category.set("Temperature");  
            temperature.set(temp);  
            context.write(category, temperature);  
  
            category.set("WindSpeed");  
            windSpeed.set(wind);  
            context.write(category, windSpeed);  
  
            category.set("DewPoint");  
            dewPoint.set(dew);  
            context.write(category, dewPoint);  
        }  
    }  
  
    public static class FloatAverageReducer extends Reducer<Text, FloatWritable, Text, FloatWritable> {  
        private FloatWritable result = new FloatWritable();  
  
        public void reduce(Text key, Iterable<FloatWritable> values, Context context)  
                throws IOException, InterruptedException {  
            float sum = 0;  
            int count = 0;  
            for (FloatWritable val : values) {  
                sum += val.get();  
                count++;  
            }  
            float avg = sum / count;  
            result.set(avg);  
            context.write(key, result);  
        }  
    }  
  
    public static void main(String[] args) throws Exception {  
        Configuration conf = new Configuration();  
        Job job = Job.getInstance(conf, "weather data average");  
        job.setJarByClass(weathermap.class);  
  
        job.setMapperClass(TokenizerMapper.class);  
        job.setReducerClass(FloatAverageReducer.class);  
  
        job.setMapOutputKeyClass(Text.class);  
        job.setMapOutputValueClass(FloatWritable.class);  
  
        job.setOutputKeyClass(Text.class);  
        job.setOutputValueClass(FloatWritable.class);  
  
        job.setInputFormatClass(TextInputFormat.class);  
        job.setOutputFormatClass(TextOutputFormat.class);  
  
        TextInputFormat.addInputPath(job, new Path(args[0]));  
        TextOutputFormat.setOutputPath(job, new Path(args[1]));  
  
        System.exit(job.waitForCompletion(true) ? 0 : 1);  
    }  
}

**Input.txt file🡪**

51.75 1006.3 71.6

54.74 1006.3 71.6

50.59 1006.3 71.6

51.67 1006.3 71.6

65.67 1006.3 71.6

55.37 1006.3 71.6

49.26 1006.3 71.6

55.44 1006.3 71.6

64.05 1006.3 71.6

68.77 1006.3 71.6

48.93 1006.3 71.6

65.37 1006.3 71.6

69.45 1006.3 71.6

52.91 1006.3 71.6

53.69 1006.3 71.6

53.3 1006.3 71.6

66.17 1006.3 71.6

53.83 1006.3 71.6

50.54 1006.3 71.6

50.27 1006.3 71.6

59.08 1006.3 71.6

53.05 1006.3 71.6

57.97 1006.3 71.6

48.23 1006.3 71.6

47.16 1003.2 53.6

69.72 1003.2 53.6

62.71 1003.2 53.6

46.34 1003.2 53.6

53.15 1003.2 53.6

64.59 1003.2 53.6

58.26 1003.2 53.6

53.27 1003.2 53.6

43.68 1003.2 53.6

65.7 1003.2 53.6

66.27 1003.2 53.6

53.05 1003.2 53.6

68.45 1003.2 53.6

49.03 1003.2 53.6

66.59 1003.2 53.6

63.12 1003.2 53.6

49.13 1003.2 53.6

62.85 1003.2 53.6

64.67 1003.2 53.6

55.73 1003.2 53.6

56.42 1003.2 53.6

53.83 1003.2 53.6

45.14 1003.2 53.6

68.18 1003.2 53.6

48.41 1008.7 62.78

55.12 1008.7 62.78

46.48 1008.7 62.78

54.99 1008.7 62.78

50.62 1008.7 62.78

55.81 1008.7 62.78

59.28 1008.7 62.78

60.55 1008.7 62.78

52.62 1008.7 62.78

62.27 1008.7 62.78

49.9 1008.7 62.78

45.12 1008.7 62.78

62.85 1008.7 62.78

50.9 1008.7 62.78

49.73 1008.7 62.78

47.28 1008.7 62.78

46.67 1008.7 62.78

49.48 1008.7 62.78

59.53 1008.7 62.78

59.49 1008.7 62.78

52.25 1008.7 62.78

57.67 1008.7 62.78

54.04 1008.7 62.78

58.76 1008.7 62.78

48.6 1020 51.98

50.54 1020 51.98

38.55 1020 51.98

50.14 1020 51.98

34.86 1020 51.98

35.38 1020 51.98

41.52 1020 51.98

42.28 1020 51.98

45.11 1020 51.98

51.01 1020 51.98

43.6 1020 51.98

50.12 1020 51.98

55.07 1020 51.98

43.24 1020 51.98

49.84 1020 51.98

44.42 1020 51.98

58.77 1020 51.98

35.32 1020 51.98

34.83 1020 51.98

51.72 1020 51.98

52.39 1020 51.98

57.54 1020 51.98

50.37 1020 51.98

51.75 1020 51.98

**ASSIGNMENT : Scala**

1)Create folder on Desktop

2)Create word\_count.scala file which is text file and write code given below

3) create input.txt file and give input as per require

4)open terminal firstly see scala -version

5)type next command🡪 spark-shell

After enter:🡪

/ \_\_/\_\_ \_\_\_ \_\_\_\_\_/ /\_\_

\_\ \/ \_ \/ \_ `/ \_\_/ '\_/

/\_\_\_/ .\_\_/\\_,\_/\_/ /\_/\\_\ version 3.3.1

/\_/

6)type command for final output🡪 :load word\_count.scala

7)Finish

**Word\_count.scala file**

**Note :🡪 check name of textfile for first line of code**

val data=sc.textFile("input.txt")

data.collect;

val splitdata = data.flatMap(line => line.split(" "));

splitdata.collect;

val mapdata = splitdata.map(word => (word,1));

mapdata.collect;

val reducedata = mapdata.reduceByKey(\_+\_);

reducedata.collect;