Lab 5) Write a Map Reduce program that mines weather data. Weather sensors collecting data every hour at many locations across the globe gather a large volume of log data, which is a good candidate for analysis with MapReduce, since it semi structured and record-oriented.

Climate change has been seeking a lot of attention since long time. The antagonistic effect of this climate is being felt in every part of the earth. There are many examples for these, such as sea levels are rising, less rainfall, increase in humidity. The propose system overcomes some issues that occurred by using other techniques. In this project we use the concept of Big data Hadoop. In the proposed architecture we are able to process offline data, which is stored in the National Climatic Data Centre (NCDC). Through this we are able to find out the maximum temperature and minimum temperature of year, and able to predict the future weather forecast. Finally, we plot the graph for the obtained MAX and MIN temperature for each moth of the particular year to visualize the temperature. Based on the previous year data weather data of coming year is predicted.

ALGORITHM:-

MAPREDUCE PROGRAM

WordCount is a simple program which counts the number of occurrences of each word in a giventext input data set. WordCount fits very well with the MapReduce programming model making ita great example to understand the Hadoop Map/Reduce programming style. Our implementation consists of three main parts:

1. Mapper
2. Reducer
3. Main program

Step-1. Write a Mapper

A Mapper overrides the –mapll function from the Class "org.apache.hadoop.mapreduce.Mapper" which provides <key, value> pairs as the input. A Mapper implementation may output
<key,value> pairs using the provided Context .

Input value of the WordCount Map task will be a line of text from the input data file and the keywould be the line number <line_number, line_of_text> . Map task outputs <word, one> for eachword in the line of text.

Pseudo-code

```
void Map (key, value){
    for each max_temp x in
        value:
        output.collect(x, 1);
}
void Map (key, value){
    for each min_temp x in
        value:
        output.collect(x, 1);
}
```

Step-2 Write a Reducer

A Reducer collects the intermediate <key,value> output from multiple map tasks and assemble asingle result. Here, the WordCount program will sum up the occurrence of each word to pairs as
<word, occurrence>.

Pseudo-code

```
void Reduce (max_temp, <list of

    value>){for each x in <list of

    value>:

        sum+=x;

    final_output.collect(max_tem

    p, sum);

}
void Reduce (min_temp, <list of

    value>){for each x in <list of

    value>:

        sum+=x;

    final_output.collect(min_tem

    p, sum);

}
```

3. Write Driver

The Driver program configures and run the MapReduce job. We use the main program to perform basic configurations such as:

Job Name : name of this Job

Executable (Jar) Class: the main executable class. For here, WordCount.

Mapper Class: class which overrides the "map" function. For here, Map.

Reducer: class which override the "reduce" function. For here, Reduce.

Output Key: type of output key. For here, Text.

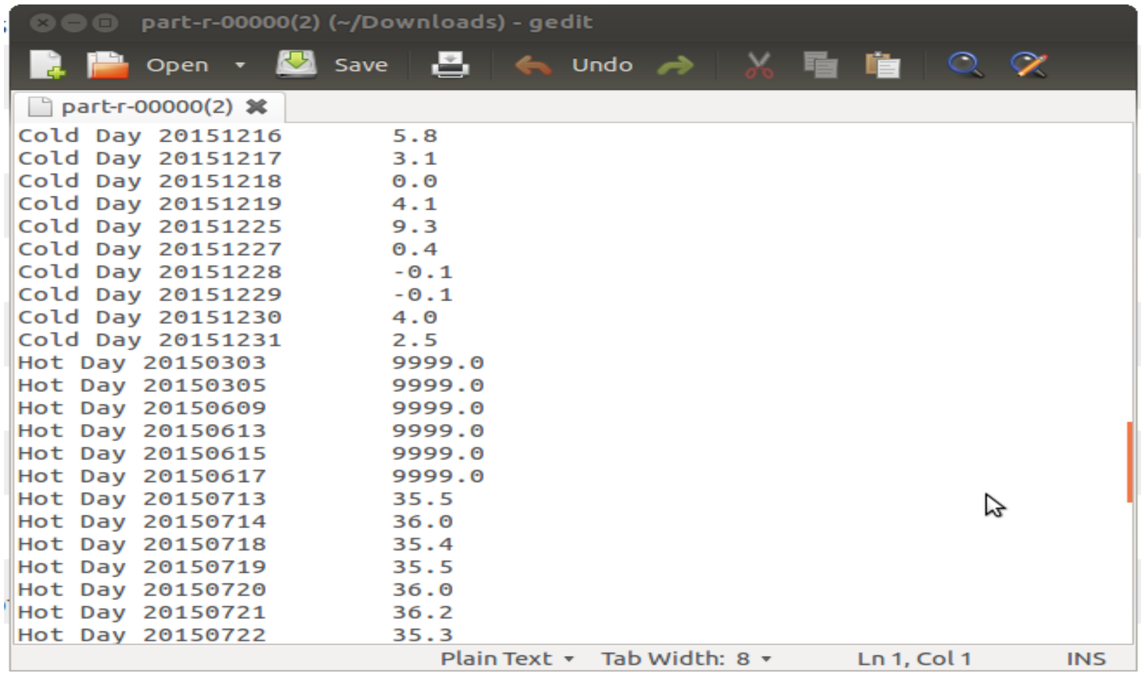
Output Value: type of output value. For here, IntWritable.

File Input Path File Output Path

INPUT:-

Set of Weather Data over the years

OUTPUT: -



```
Cold Day 20151216	5.8
Cold Day 20151217	3.1
Cold Day 20151218	0.0
Cold Day 20151219	4.1
Cold Day 20151225	9.3
Cold Day 20151227	0.4
Cold Day 20151228	-0.1
Cold Day 20151229	-0.1
Cold Day 20151230	4.0
Cold Day 20151231	2.5
Hot Day 20150303	9999.0
Hot Day 20150305	9999.0
Hot Day 20150609	9999.0
Hot Day 20150613	9999.0
Hot Day 20150615	9999.0
Hot Day 20150617	9999.0
Hot Day 20150713	35.5
Hot Day 20150714	36.0
Hot Day 20150718	35.4
Hot Day 20150719	35.5
Hot Day 20150720	36.0
Hot Day 20150721	36.2
Hot Day 20150722	35.3
```