Appendix A: Hadoop Commands:

- ssh localhost
- ssh-keygen -t rsa -P “”
- cat /Users/hadoop/.ssh/id_rsa.pub >> /Users/hadoop/.ssh/authorized_keys
- hadoop$tar -xzvf hadoop-*
- bin/hdfs namenode -format   //remove namenode
- rm -r /tmp/hadoop-hadoop/dfs/data/current   // to remove the data node
- *Upload input file:* In localhost:9870
- *To add folder:* bin/hdfs dfs -mkdir /user
- bin/hadoop jar WordCount.jar WordCount /user /op2
- bin/hdfs dfs -cat /op2/part-r-00000
- *Inside sbin folder:*   ./start-all.sh ./stop-all.sh

Appendix B: Implemented Hadoop code:

Code 1:

```
import java.io.IOException;

import java.util.*;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.conf.*;

import org.apache.hadoop.io.*;

import org.apache.hadoop.mapreduce.*;

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

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

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

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

public class Mu1{

  public static class TokenizerMapper extends Mapper<Object, Text, Text, FloatWritable>{

  private static int count=0;

/*

input: dataset

*/

 public void map(Object key, Text value, Context context) throws IOException,
InterruptedException {

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

```
String[] w=line.split(" ");
int a=0,b=0,c=0;
int s=0;
Integer a1,b1,c1;
a1=Integer.valueOf(w[0]);
b1=Integer.valueOf(w[2]);
c1=Integer.valueOf(w[4]);
a=a1.intValue();
b=b1.intValue();
c=c1.intValue();
s=a+b+c;
Integer s1=new Integer(s);
float sf=s1.floatValue();
count++;
context.write(new Text(String.valueOf(sf+count)), new FloatWritable((float) count));
    }
  }
/*
output:
key: sum+ number
value: number
*/
public static class IntSumReducer extends Reducer<Text,FloatWritable,Text,FloatWritable>
{
public static int c=24;
public static float c1=0;
    public void reduce(Text key, Iterable<FloatWritable> values,Context context) throws
IOException, InterruptedException {
c--;
c1++;
FloatWritable cf=new FloatWritable((float)(c/24.0f));
```

```
// to calculate the m(n) lowest value
String kw=key.toString();
String[] w=kw.split(" ");
float k1=Float.valueOf(w[0]);
k1=k1-c1;
context.write(new Text(String.valueOf(k1)),cf);
}
  }
/*
output:
key: xsum
value: mu
*/

public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();
    Job job = Job.getInstance(conf, "Mu");
    job.setJarByClass(Mu1.class);
    job.setMapperClass(TokenizerMapper.class);
    job.setCombinerClass(IntSumReducer.class);
    job.setReducerClass(IntSumReducer.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(FloatWritable.class);
    FileInputFormat.addInputPath(job,new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));
    System.exit(job.waitForCompletion(true) ? 0 : 1);
  }
}
```

Code 2:

```
import java.io.IOException;

import java.util.*;

import java.lang.*;

import java.util.*;

import java.io.RandomAccessFile;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.conf.*;

import org.apache.hadoop.io.*;

import org.apache.hadoop.mapreduce.*;

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

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

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

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

public class Tou1 {

  public static class TokenizerMapper extends Mapper<LongWritable,Text, IntWritable,
FloatWritable>{

   public static float di=0.0f;

   public static int index=0;

/*

input:

key: xsum

value:mu

*/

        public void map(LongWritable key, Text value, Context context) throws
IOException, InterruptedException {

                String a,b;

                float min=10000.0f;

                float s1=0.0f;

                int cluster=0;
                RandomAccessFile areader=new
RandomAccessFile("/home/hadoop/hadoop/project/tou/mu.txt","r");
                RandomAccessFile breader=new
RandomAccessFile("/home/hadoop/hadoop/project/tou/mu.txt","r");
                while(!(a=areader.readLine()).equals(null))
          {
                float m=Float.valueOf(a);
                 while(!(b=breader.readLine()).equals(null))
                {
                  float mn=Float.valueOf(b);
                  float dd=Math.abs(m-mn);
                  s1=s1+dd;//find the value of p
                  index++;//cluster centroid id
                   if(min>s1)
                   {
                         min=s1; // (pi(F))
                         cluster=index;// (xi(F)) first cluster centroid
                   }
                }//breader
                breader.seek(0);
          }//areader
         breader.close();
         areader.close();
         float min1;
          // to calculate di
          RandomAccessFile areader1=new
RandomAccessFile("/home/hadoop/hadoop/project/tou/xsum.txt","r");
          areader1.seek((int)min);
          min1=Float.valueOf(areader1.readLine());
          areader1.seek(0);
```

```
                    while(!((a=areader1.readLine()).equals(null)))
                    {
                     float mn=(Float.valueOf(a));
                     di=(mn-min1)*(mn-min1);
                     context.write(new IntWritable(1), new FloatWritable(di));
                    }
            areader1.close();
     }//map
  }//mapper

/*
mapper output:
key: added sum of all columns
value: di
*/

 public static class IntSumReducer extends
Reducer<IntWritable,FloatWritable,IntWritable,FloatWritable> {
  public static float dbar=0.0f;
  public static int count=24;
  public static float tou1=0.0f;
  public static float tou2=0.0f;
  LinkedList db = new LinkedList();
   public void reduce(IntWritable key,Iterable<FloatWritable> values,Context context) throws
IOException, InterruptedException {
    float r=0.0f;
    for (FloatWritable value : values)
          {
                    float s=value.get();
                    dbar+=s;//dbar
                    db.add(s);
          }
                    dbar/=count;
                    float dd=0.0f;
                    Iterator<Float> itr=db.iterator();
                    while(itr.hasNext())
                    {
                              dd=itr.next();
                              tou1+=(dd)-dbar;
                    }
                    tou1/=count;
                    tou2=tou1*1.5f;
                    context.write(new IntWritable((int)Math.abs(tou1)) ,new
FloatWritable(Math.abs(tou2)));
    }//reduce
  }//reducer

/*
output of reduce
key: tou1
value: tou2
*/

public static void main(String[] args) throws Exception {
          Configuration conf = new Configuration();
          Job job = Job.getInstance(conf, "Tou1");
          job.setJarByClass(Tou1.class);
          job.setMapperClass(TokenizerMapper.class);
          job.setCombinerClass(IntSumReducer.class);
          job.setReducerClass(IntSumReducer.class);
          job.setOutputKeyClass(IntWritable.class);
          job.setOutputValueClass(FloatWritable.class);
```

```
            FileInputFormat.addInputPath(job,new Path(args[0]));
            FileOutputFormat.setOutputPath(job, new Path(args[1]));
            System.exit(job.waitForCompletion(true) ? 0 : 1);
  }
}
```

Code 3:

```
import java.io.IOException;
import java.io.RandomAccessFile;
import java.util.*;
import java.lang.*;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.conf.*;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapreduce.*;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextautputFormat;
public class Mountain {
public static class TokenizerMapper extends Mapper<Object, Text,
IntWritable,FloatWritable>    {
IntWritable one=new IntWritable(1);
int test=0;
 static float maxM=0.0f;


/*
Input of map
key: tou1
value: tou2
*/
```

```
 public void map(LongWritable key, Text value, Context context) throws IOException,
InterruptedException {

float temp=0.0f;

try{

float xone=0.0f;

String a;

RandomAccessFile x=new
RandomAccessFile("/home/hadoop/hadoop/project/tau.txt","r");//file with tau values

a=x.readLine();

float tau1=Float.valueOf(a);

a=x.readLine();

float tau2=Float.valueOf(a);

float inter1,inter2=0.0f;

float inter3=0.0f;

float inter4=0.0f;

float diff=0.0f;

float tau12=(tau1/2)*(tau1/2);//deno l

float tau22=(tau2/2)*(tau2/2);//deno l-1

 LinkedList mi = new LinkedList();

 LinkedList mil = new LinkedList();//centroid value

float xs=0.0f;//difference value

float xx=1.0f;//condition

String b;

int d=0;

while(xx!=0.0f)

{

RandomAccessFile areader=new
RandomAccessFile("/home/hadoop/hadoop/project/xsum.txt","r");//dataset x value

RandomAccessFile breader=new
RandomAccessFile("/home/hadoop/hadoop/project/xsum.txt","r");

while(!((a=areader.readLine()).equals(null)))

  {
```

```
  float xi=Float.valueOf(a);
while(!((b=breader.readLine()).equals(null)))
{
float xj=Float.valueOf(b);
diff=(xi-xj)*(xi-xj);//numo
inter1=(float)Math.exp(-(diff/tau12));//exp value
inter2+=inter1;
if(maxM<inter2)
{
if(temp!=xone)
{
  test=1;
}
temp=xone;
maxM=inter2;
xone=xi;
}
}
breader.seek(0);
mi.add(inter2);//Mi values in linked list l
inter2=0.0f;
}//while
breader.close();
areader.close();
//update mountain
float mI=0.0f;
RandomAccessFile areader1=new
RandomAccessFile("/home/hadoop/hadoop/project/xsum.txt","r");//dataset x value
RandomAccessFile breader1=new
RandomAccessFile("/home/hadoop/hadoop/project/xsum.txt","r");
while(!((a=areader1.readLine()).equals(null)))
```

```
{
float xi=Float.valueOf(a);
while(!((b=breader1.readLine()).equals(null)))
{
float xj=Float.valueOf(b);
diff=(xi-xj)*(xi-xj);//numo
inter3=(float)Math.exp(-(diff/tau12));//exp value
Iterator<Float> itr=mi.iterator();
while(itr.hasNext()){
mI=(itr.next());
inter4=mI*inter3;//left side value
mil.add(inter4);//Mi value Linked list l+1
}
}
breader1.seek(0);
}//while
breader1.close();
areader1.close();
float mm=0.0f;
float mm1=0.0f;
 Iterator<Float> itr1=mi.iterator();
 Iterator<Float> itr2=mil.iterator();
while(itr1.hasNext()){
mm=(itr1.next());
mm1=(itr2.next());
xs=mm-mm1;
if(xs>(0.6*mm))
xx=0.0f;
}
}
```

```
if(test==1)

{

  context.write(new IntWritable(one),new FloatWritable(xone));

  test=0;

}

}

catch(Exception e)

{

System.out.println(e);

}

}//map

}//mapper

/*

Output of reduce:

Key: No: of centroid

Value: Centroid

*/

public static class IntSumReducer extends
Reducer<IntWritable,FloatWritable,IntWritable,FloatWritable> {

  public static int cen=0;

    public void reduce(IntWritable key, Iterable<FloatWritable> values,Context context)
throws IOException, InterruptedException {

      for (FloatWritable value : values)

 {

            cen++;

            context.write(new IntWritable(cen),value);

  }

}//reduce

  }//reducer


public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();
    Job job = Job.getInstance(conf, "Mountain");
    job.setJarByClass(Mountain.class);
    job.setMapperClass(TokenizerMapper.class);
    job.setCombinerClass(IntSumReducer.class);
    job.setReducerClass(IntSumReducer.class);
    job.setautputKeyClass(IntWritable.class);
    job.setautputValueClass(FloatWritable.class);
    FileInputFormat.addInputPath(job,new Path(args[0]));
    FileOutputFormat.setautputPath(job, new Path(args[1]));
    System.exit(job.waitForCompletion(true) ? 0 : 1);
  }
}
```