Map.java:

```
Package matrix;

Import org.apache.hadoop.conf.*;

Import org.apache.hadoop.io.LongWritable;

Import org.apache.hadoop.io.Text;

//import org.apache.hadoop.mapreduce.Mapper;

Import java.io.IOException;

Public class Map extends org.apache.hadoop.mapreduce.Mapper<LongWritable, Text,
Text,

Text>

{

@Override

Public void map(LongWritable key, Text value, Context context)

Throws IOException, InterruptedException {
```

```
Configuration conf = context.getConfiguration();

Int m = Integer.parseInt(conf.get(“m”));

Int p = Integer.parseInt(conf.get(“p”));

String line = value.toString();

// (M, I, j, Mij);

String[] indicesAndValue = line.split(“,”);

Text outputKey = new Text();

Text outputValue = new Text();

If (indicesAndValue[0].equals(“M”)) {

For (int k = 0; k < p; k++) {

outputKey.set(indicesAndValue[1] + “,” + k);
// outputKey.set(I,k);

outputValue.set(indicesAndValue[0] + “,” + indicesAndValue[2]
+ “,” + indicesAndValue[3]);

// outputValue.set(M,j,Mij);

Context.write(outputKey, outputValue);

}

} else {

// (N, j, k, Njk);

For (int I = 0; I < m; i++) {

outputKey.set(I + “,” + indicesAndValue[2]); outputValue.set(“N,” +

indicesAndValue[1] + “,”

+ indicesAndValue[3]); context.write(outputKey, outputValue);

}

}

}
```

```
}

MatrixMultiply.java

Package matrix;

Import org.apache.hadoop.conf.*;

Import org.apache.hadoop.fs.Path;

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 MatrixMultiply {

Public static void main(String[] args) throws Exception { if (args.length != 2) {
```

```
System.err.println(“Usage: MatrixMultiply <in_dir> <out_dir>”);

System.exit(2);

}

Configuration conf = new Configuration();

Conf.set(“m”, “1000”);

Conf.set(“n”, “100”);

Conf.set(“p”, “1000”);

@SuppressWarnings(“deprecation”)

Job job = new Job(conf, “MatrixMultiply”);

Job.setJarByClass(MatrixMultiply.class);

Job.setOutputKeyClass(Text.class);

Job.setOutputValueClass(Text.class);

Job.setMapperClass(Map.class);
Job.setReducerClass(Reduce.class);

Job.setInputFormatClass(TextInputFormat.class);

Job.setOutputFormatClass(TextOutputFormat.class);

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

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

Job.waitForCompletion(true);

}

}
```

Reduce.java

```
Package matrix;

Import org.apache.hadoop.io.Text;

// import org.apache.hadoop.mapreduce.Reducer;

Import java.io.IOException;
Import java.util.HashMap;

Public class Reduce

Extends org.apache.hadoop.mapreduce.Reducer<Text, Text, Text, Text> { @Override

Public void reduce(Text key, Iterable<Text> values, Context context)

Throws IOException, InterruptedException {

String[] value;
//key=(I,k),

//Values = [(M/N,j,V/W),..]

HashMap<Integer, Float> hashA = new HashMap<Integer, Float>(); HashMap<Integer,

Float> hashB = new HashMap<Integer, Float>(); for (Text val : values) {

Value = val.toString().split(“,”);

If (value[0].equals(“M”)) {

hashA.put(Integer.parseInt(value[1]), Float.parseFloat(value[2])); } else {

hashB.put(Integer.parseInt(value[1]), Float.parseFloat(value[2]));
}

}

Int n = Integer.parseInt(context.getConfiguration().get(“n”));

Float result = 0.0f;

Float m_ij;

Float n_jk;

For (int j = 0; j < n; j++) {

M_ij = hashA.containsKey(j) ? hashA.get(j) : 0.0f; n_jk = hashB.containsKey(j) ?

hashB.get(j) : 0.0f; result += m_ij * n_jk;

}

If (result != 0.0f) {

Context.write(null,

New Text(key.toString() + “,” + Float.toString(result)));
```

```
}


}


}
```