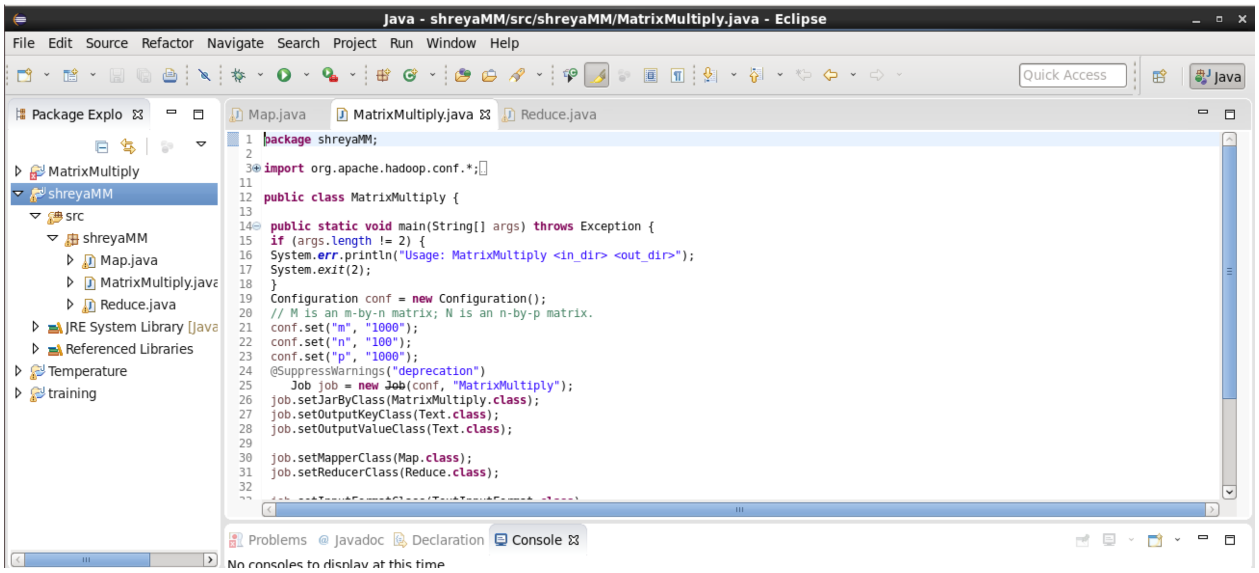
EXPERIMENT 8

Name	Shreya Shetty
UID	20191410059
Batch	A
Class	TE IT
Subject	BDA

AIM: Implement algorithms in Map-reduce on Relational Algebra (Matrix Multiplication).

CODE:

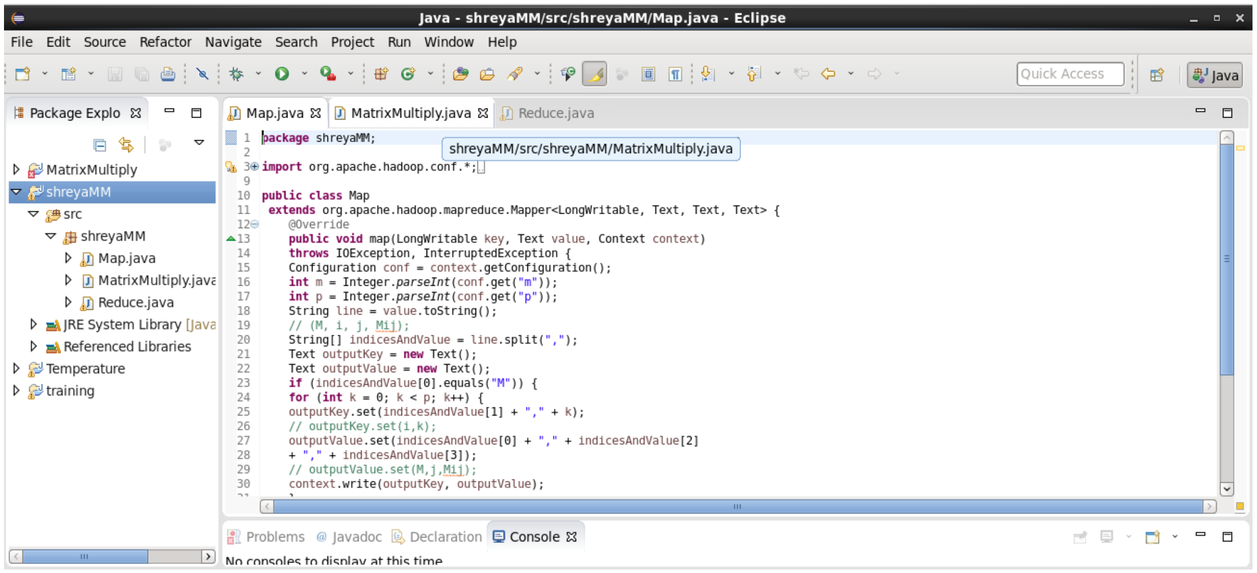
Driver Code (MatrixMultiply.java)



```
package shreyaMM;

import org.apache.hadoop.conf.*;

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();
 // M is an m-by-n matrix; N is an n-by-p matrix.
 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);

```

```
No consoles to display at this time
```

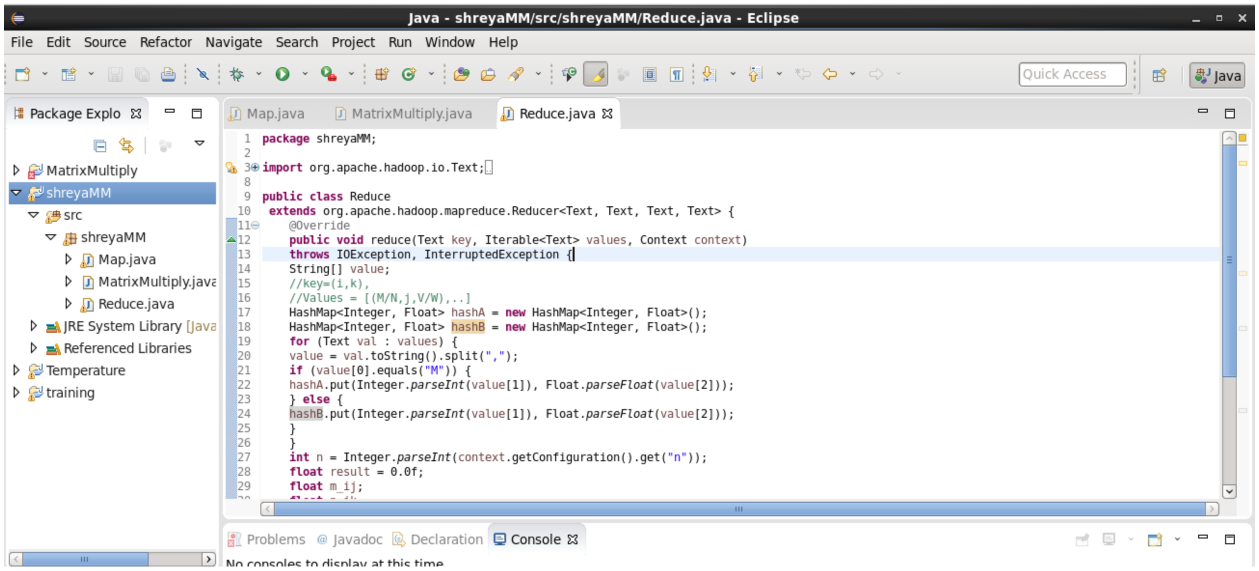
Mapper Code (Map.java)



```java
package shreyaMM;

import org.apache.hadoop.conf.*;

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);
```

```
No consoles to display at this time
```

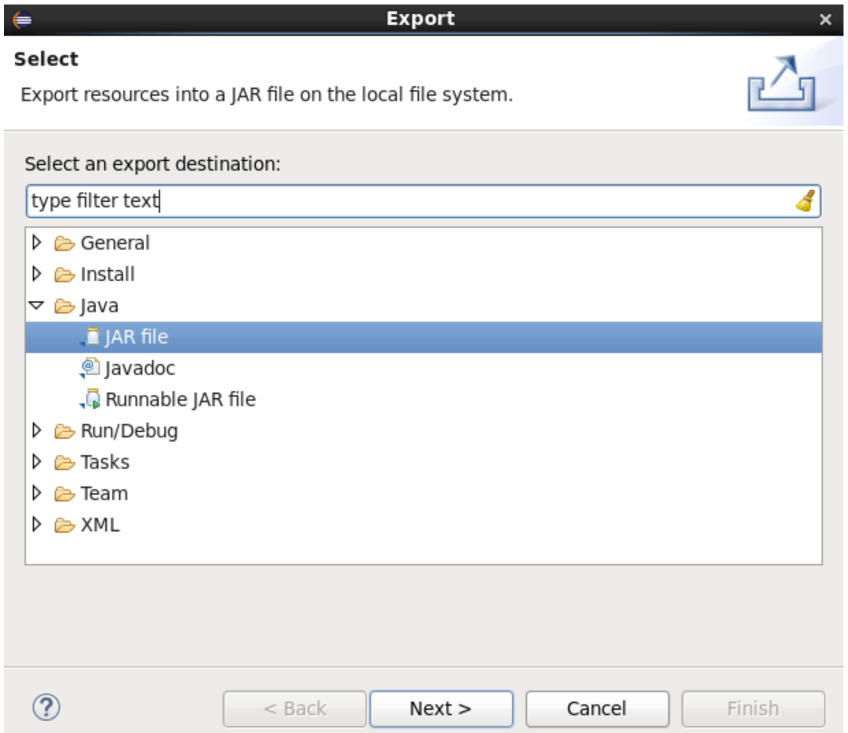
Reducer Code (Reduce.java)



```java
package shreyaMM;

import org.apache.hadoop.io.Text;

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;
```

```
No consoles to display at this time
```

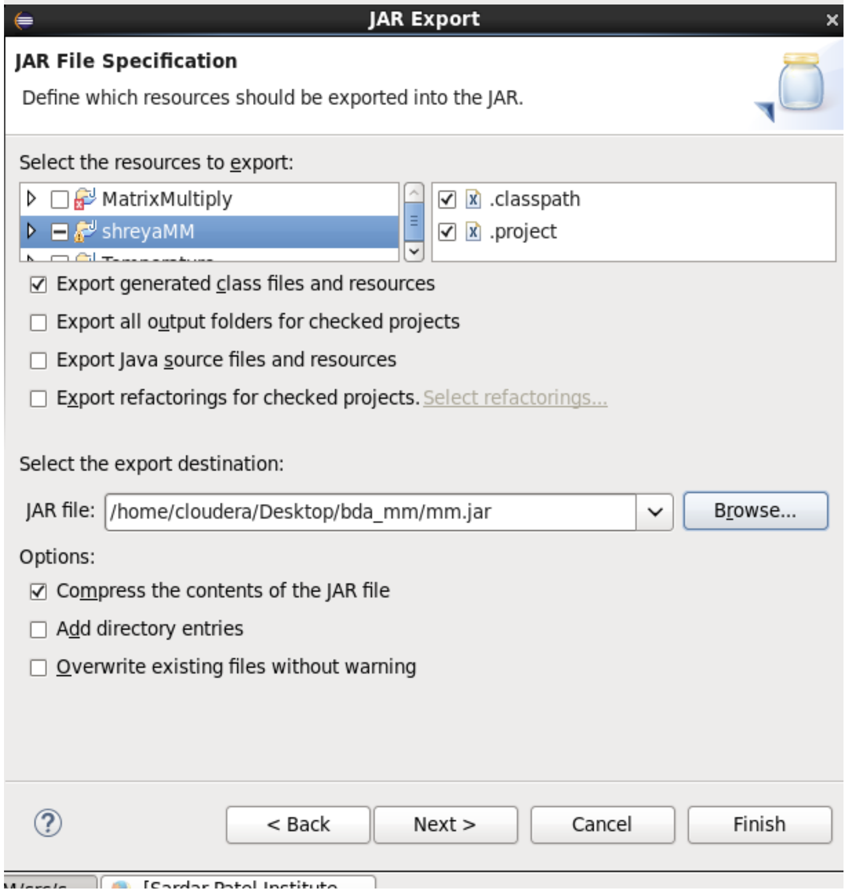
EXPORTING THE JAR FILE OF PROJECT AND SAVING IT TO LOCAL SYSTEM:



Export resources into a JAR file on the local file system.

Select an export destination:

type filter text



Define which resources should be exported into the JAR.

Select the resources to export:

☑ .classpath
☑ .project

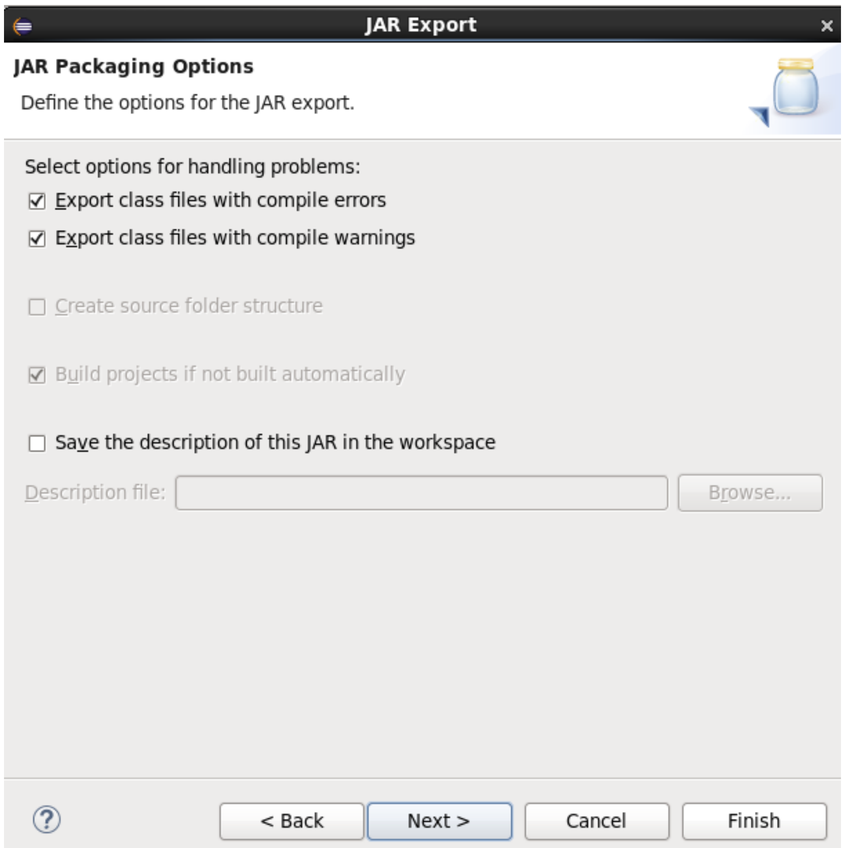
☑ Export generated class files and resources
☐ Export all output folders for checked projects
☐ Export Java source files and resources
☐ Export refactorings for checked projects. Select refactorings...

Select the export destination:

JAR file: /home/cloudera/Desktop/bda_mm/mm.jar

Options:

☑ Compress the contents of the JAR file
☐ Add directory entries
☐ Overwrite existing files without warning



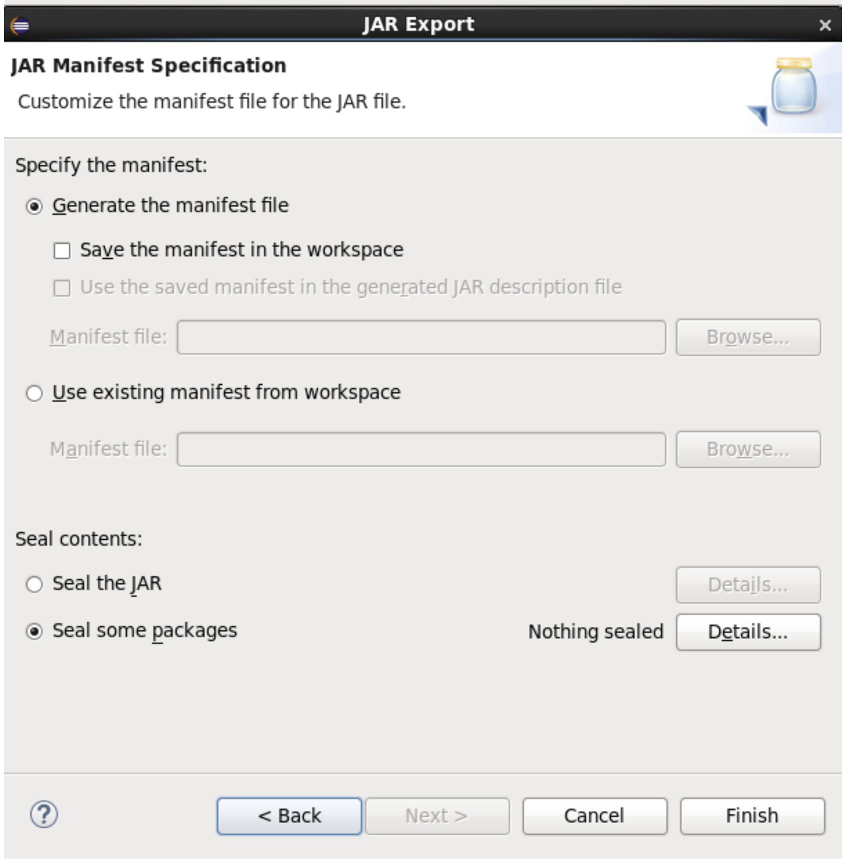
JAR Packaging Options

Define the options for the JAR export.

Select options for handling problems:

- [✓] Export class files with compile errors
- [✓] Export class files with compile warnings
- [ ] Create source folder structure
- [✓] Build projects if not built automatically
- [ ] Save the description of this JAR in the workspace

Description file:



JAR Manifest Specification

Customize the manifest file for the JAR file.

Specify the manifest:

- [●] Generate the manifest file
  - [ ] Save the manifest in the workspace
  - [ ] Use the saved manifest in the generated JAR description file
  - Manifest file:
- [ ] Use existing manifest from workspace
  - Manifest file:

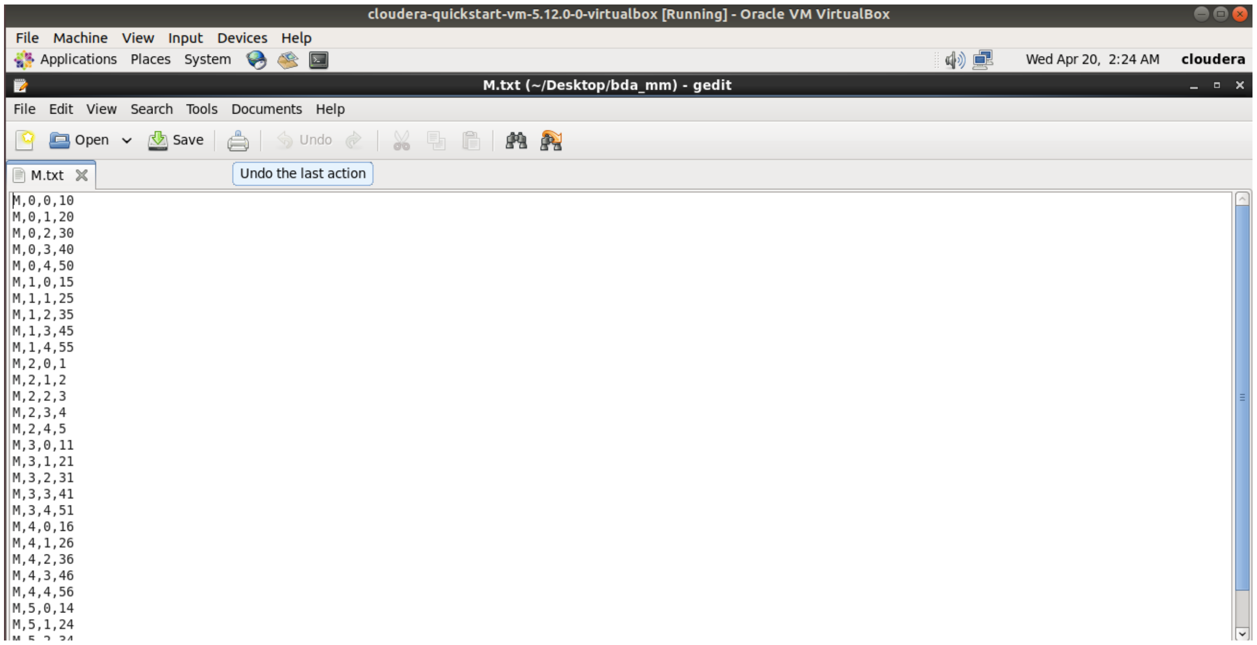
Seal contents:

- [ ] Seal the JAR
- [●] Seal some packages — Nothing sealed

INPUT MATRICES:

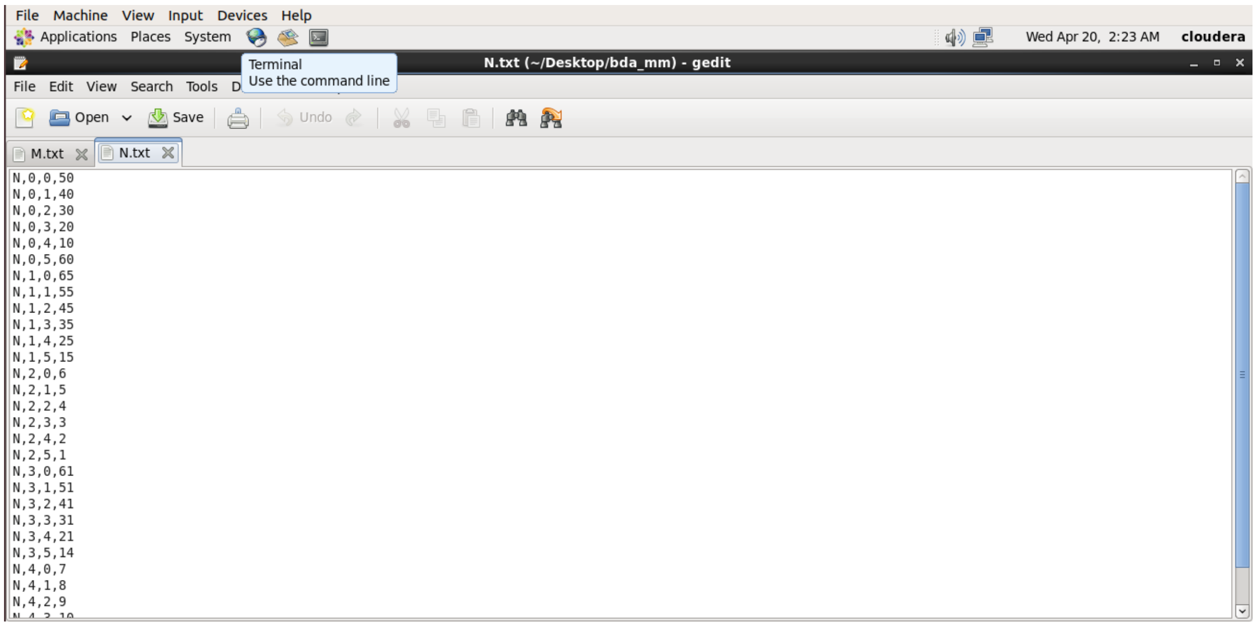
Here, the 1st character represents the matrix which it belongs to, while the 2nd and 3rd character represent the ith and jth index of the matrix and the last one represents the value at that particular (i,j)th index.

M.txt


cloudera-quickstart-vm-5.12.0-0-virtualbox [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications Places System
Wed Apr 20, 2:24 AM
cloudera
M.txt (~/Desktop/bda_mm) - gedit
File Edit View Search Tools Documents Help
Open Save Undo
M.txt
Undo the last action

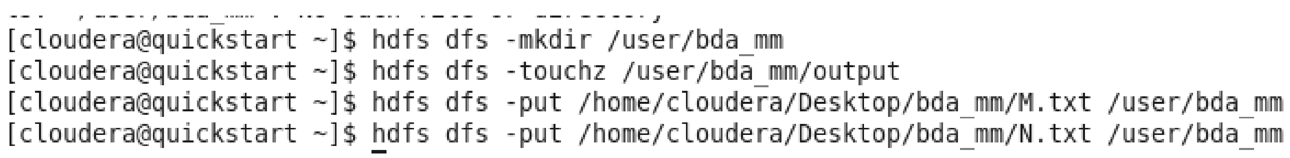
```
M,0,0,10
M,0,1,20
M,0,2,30
M,0,3,40
M,0,4,50
M,1,0,15
M,1,1,25
M,1,2,35
M,1,3,45
M,1,4,55
M,2,0,1
M,2,1,2
M,2,2,3
M,2,3,4
M,2,4,5
M,3,0,11
M,3,1,21
M,3,2,31
M,3,3,41
M,3,4,51
M,4,0,16
M,4,1,26
M,4,2,36
M,4,3,46
M,4,4,56
M,5,0,14
M,5,1,24
```

N.txt


File Machine View Input Devices Help
Applications Places System
Wed Apr 20, 2:23 AM
cloudera
Terminal
Use the command line
N.txt (~/Desktop/bda_mm) - gedit
File Edit View Search Tools D
Open Save Undo
M.txt
N.txt

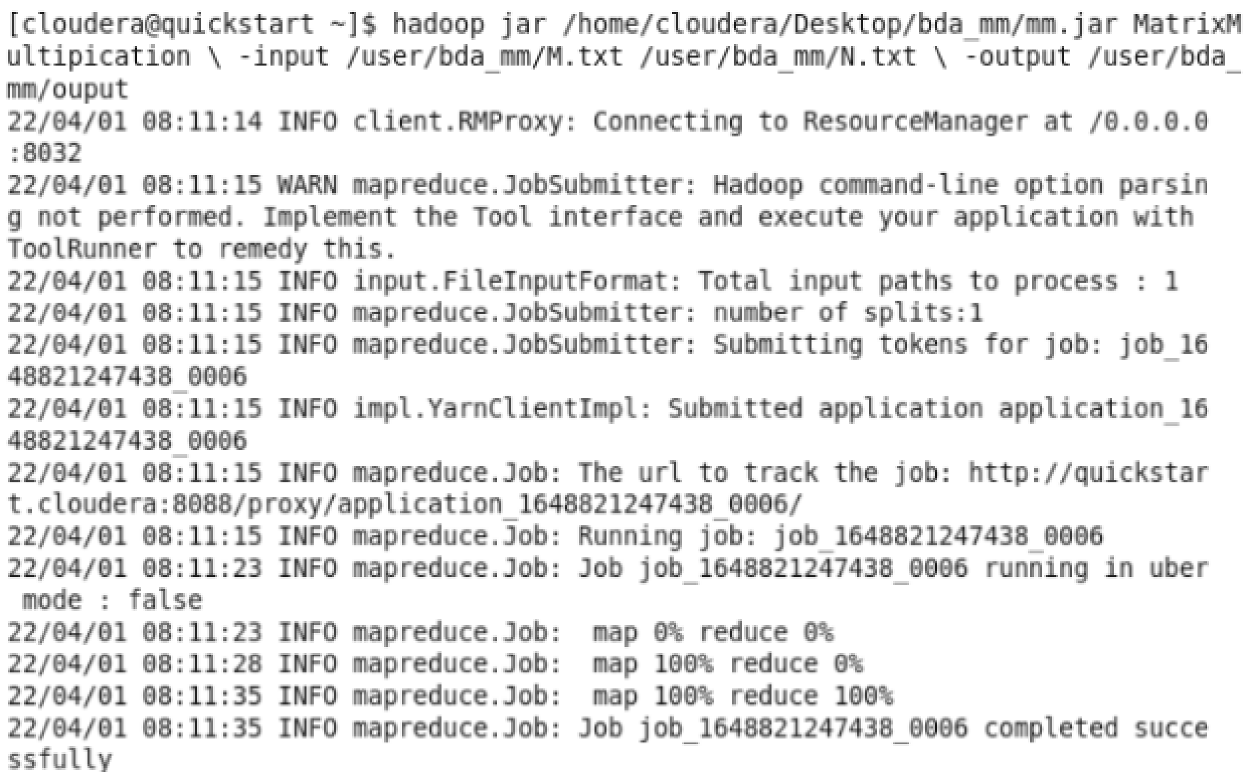
```
N,0,0,50
N,0,1,40
N,0,2,30
N,0,3,20
N,0,4,10
N,0,5,60
N,1,0,65
N,1,1,55
N,1,2,45
N,1,3,35
N,1,4,25
N,1,5,15
N,2,0,6
N,2,1,5
N,2,2,4
N,2,3,3
N,2,4,2
N,2,5,1
N,3,0,61
N,3,1,51
N,3,2,41
N,3,3,31
N,3,4,21
N,3,5,14
N,4,0,7
N,4,1,8
N,4,2,9
```

CREATING AN OUTPUT FILE IN HDFS AND PUTTING BOTH THE INPUT FILES IN HDFS



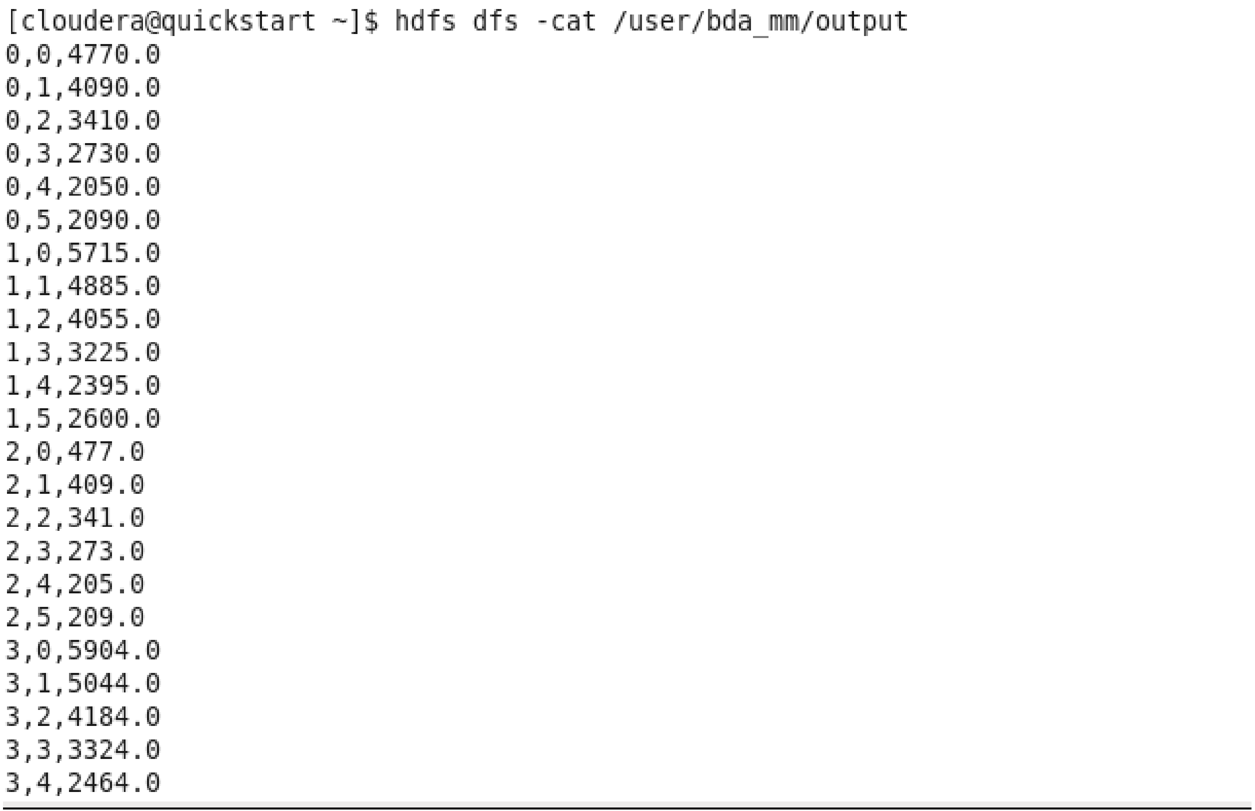
```
[cloudera@quickstart ~]$ hdfs dfs -mkdir /user/bda_mm
[cloudera@quickstart ~]$ hdfs dfs -touchz /user/bda_mm/output
[cloudera@quickstart ~]$ hdfs dfs -put /home/cloudera/Desktop/bda_mm/M.txt /user/bda_mm
[cloudera@quickstart ~]$ hdfs dfs -put /home/cloudera/Desktop/bda_mm/N.txt /user/bda_mm
```

RUNNING THE JAR FILE CREATED WITH THE INPUT MATRICES AND STORING THE OUTPUT IN OUTPUT FOLDER:

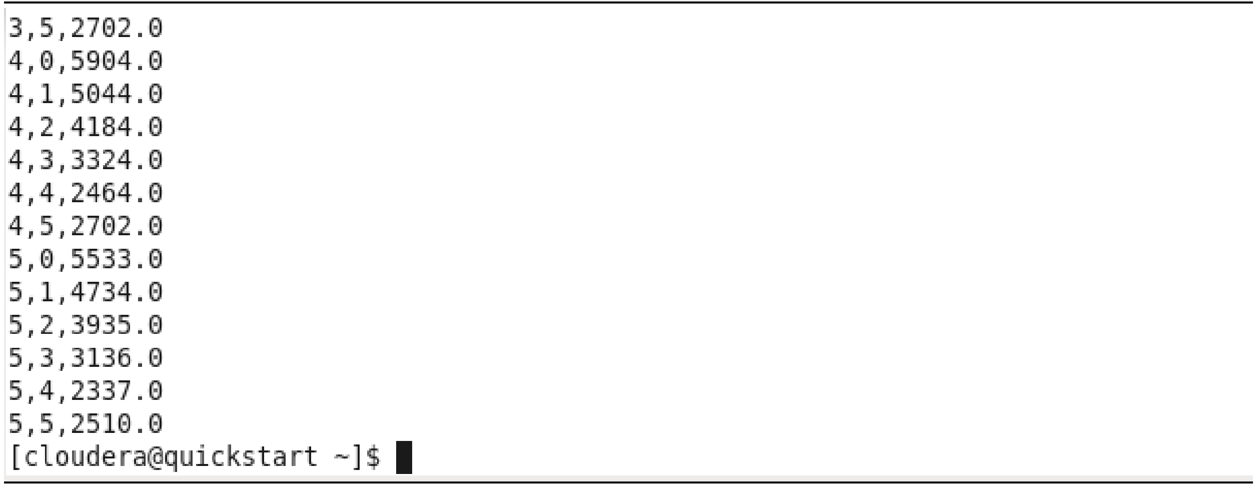


```
[cloudera@quickstart ~]$ hadoop jar /home/cloudera/Desktop/bda_mm/mm.jar MatrixM
ultipication \ -input /user/bda_mm/M.txt /user/bda_mm/N.txt \ -output /user/bda_
mm/ouput
22/04/01 08:11:14 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0
:8032
22/04/01 08:11:15 WARN mapreduce.JobSubmitter: Hadoop command-line option parsin
g not performed. Implement the Tool interface and execute your application with
ToolRunner to remedy this.
22/04/01 08:11:15 INFO input.FileInputFormat: Total input paths to process : 1
22/04/01 08:11:15 INFO mapreduce.JobSubmitter: number of splits:1
22/04/01 08:11:15 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_16
48821247438_0006
22/04/01 08:11:15 INFO impl.YarnClientImpl: Submitted application application_16
48821247438_0006
22/04/01 08:11:15 INFO mapreduce.Job: The url to track the job: http://quickstar
t.cloudera:8088/proxy/application_1648821247438_0006/
22/04/01 08:11:15 INFO mapreduce.Job: Running job: job_1648821247438_0006
22/04/01 08:11:23 INFO mapreduce.Job: Job job_1648821247438_0006 running in uber
 mode : false
22/04/01 08:11:23 INFO mapreduce.Job:  map 0% reduce 0%
22/04/01 08:11:28 INFO mapreduce.Job:  map 100% reduce 0%
22/04/01 08:11:35 INFO mapreduce.Job:  map 100% reduce 100%
22/04/01 08:11:35 INFO mapreduce.Job: Job job_1648821247438_0006 completed succe
ssfully
```

OUTPUT

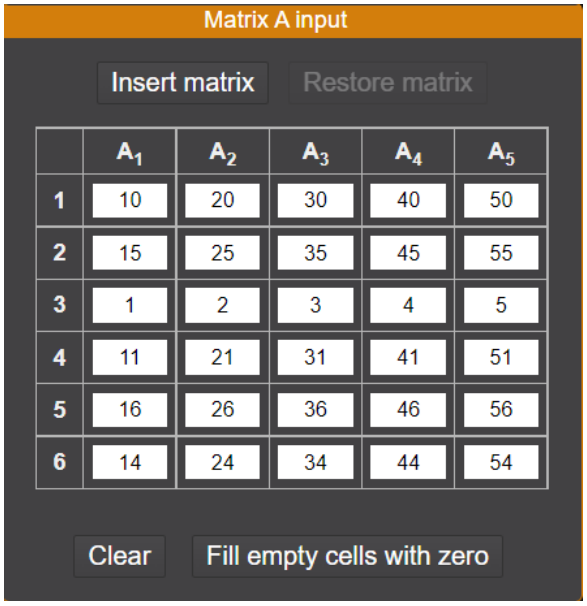


```
[cloudera@quickstart ~]$ hdfs dfs -cat /user/bda_mm/output
0,0,4770.0
0,1,4090.0
0,2,3410.0
0,3,2730.0
0,4,2050.0
0,5,2090.0
1,0,5715.0
1,1,4885.0
1,2,4055.0
1,3,3225.0
1,4,2395.0
1,5,2600.0
2,0,477.0
2,1,409.0
2,2,341.0
2,3,273.0
2,4,205.0
2,5,209.0
3,0,5904.0
3,1,5044.0
3,2,4184.0
3,3,3324.0
3,4,2464.0
```

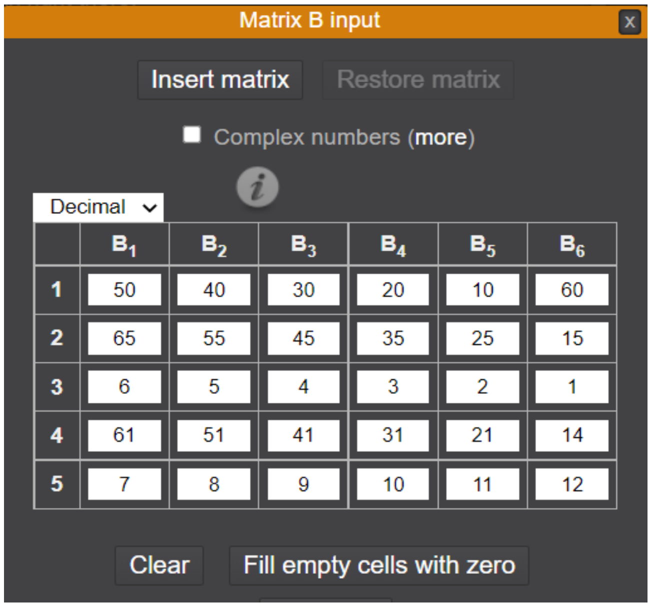


```
3,5,2702.0
4,0,5904.0
4,1,5044.0
4,2,4184.0
4,3,3324.0
4,4,2464.0
4,5,2702.0
5,0,5533.0
5,1,4734.0
5,2,3935.0
5,3,3136.0
5,4,2337.0
5,5,2510.0
[cloudera@quickstart ~]$ 
```

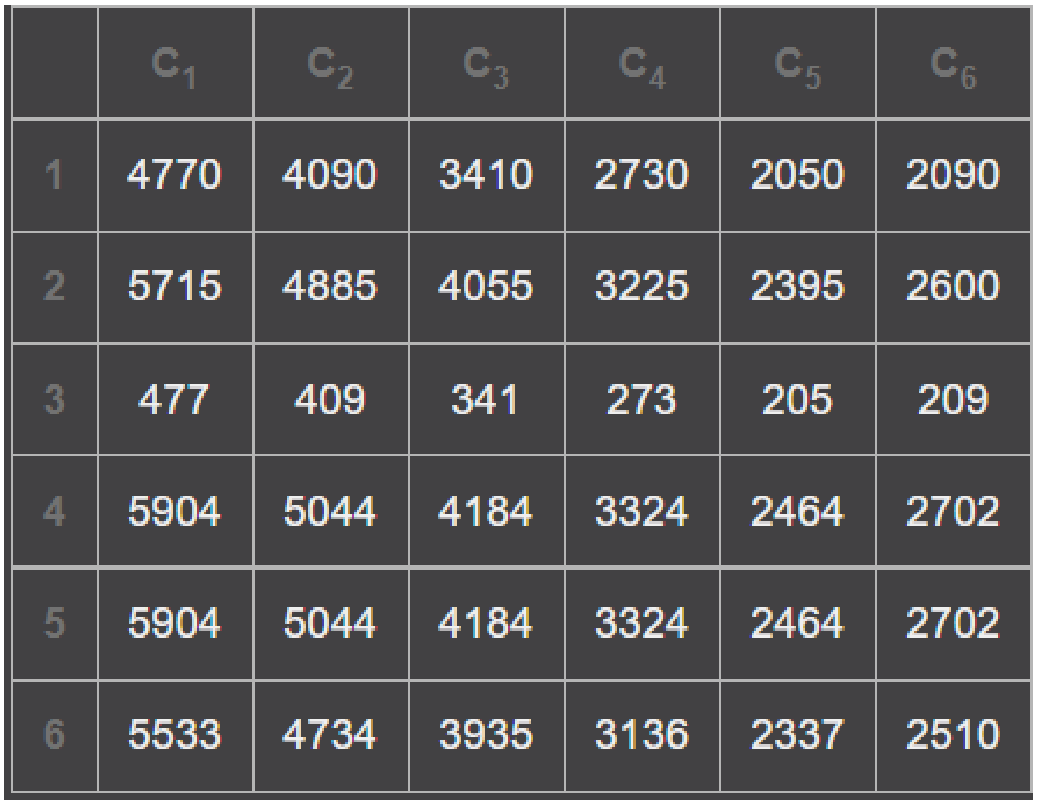
VERIFYING THE OUTPUT



	A_1	A_2	A_3	A_4	A_5
1	10	20	30	40	50
2	15	25	35	45	55
3	1	2	3	4	5
4	11	21	31	41	51
5	16	26	36	46	56
6	14	24	34	44	54



	B_1	B_2	B_3	B_4	B_5	B_6
1	50	40	30	20	10	60
2	65	55	45	35	25	15
3	6	5	4	3	2	1
4	61	51	41	31	21	14
5	7	8	9	10	11	12



	C_1	C_2	C_3	C_4	C_5	C_6
1	4770	4090	3410	2730	2050	2090
2	5715	4885	4055	3225	2395	2600
3	477	409	341	273	205	209
4	5904	5044	4184	3324	2464	2702
5	5904	5044	4184	3324	2464	2702
6	5533	4734	3935	3136	2337	2510

CONCLUSION: In this experiment, I have successfully performed the matrix multiplication in hadoop using the concept of One Step Mapreduce