# Assignment 2

Student Name: Akhil Kumar Gour
SJSU ID: 012455586

## Source Code

### Driver 1:

```
package assign_2;

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

public class driver1 {

	@SuppressWarnings("deprecation")
	public static void main(String[] args) throws Exception {
		String temp_path = "temp";
		Configuration conf = new Configuration();
		Job job1 = new Job(conf, "Conferences");
		job1.setJarByClass(driver1.class);
		job1.setMapperClass(mapper1.class);
		job1.setReducerClass(reducer1.class);
		job1.setMapOutputKeyClass(Text.class);
		job1.setMapOutputValueClass(IntWritable.class);
		job1.setOutputKeyClass(Text.class);
		job1.setOutputValueClass(IntWritable.class);
		FileInputFormat.addInputPath(job1, new Path(args[0]));
		FileOutputFormat.setOutputPath(job1, new
Path(args[1]+"/1"));
		job1.waitForCompletion(true);

		Job job2 = new Job(conf, "Conferences");
		job2.setJarByClass(driver1.class);
		job2.setMapperClass(mapper2.class);
		job2.setReducerClass(reducer2.class);
		job2.setMapOutputKeyClass(Text.class);
		job2.setMapOutputValueClass(Text.class);
		job2.setOutputKeyClass(Text.class);
		job2.setOutputValueClass(Text.class);
		FileInputFormat.addInputPath(job2, new Path(args[0]));
		FileOutputFormat.setOutputPath(job2, new
Path(args[1]+"/2"));
		job2.waitForCompletion(true);


		Job job3 = new Job(conf, "Conferences");
		job3.setJarByClass(driver1.class);
```

```
           job3.setMapperClass(mapper3.class);
           job3.setReducerClass(reducer3.class);
           job3.setMapOutputKeyClass(Text.class);
           job3.setMapOutputValueClass(Text.class);
           job3.setOutputKeyClass(Text.class);
           job3.setOutputValueClass(Text.class);
           FileInputFormat.addInputPath(job3, new Path(args[0]));
           FileOutputFormat.setOutputPath(job3, new
Path(args[1]+"/3"));
           job3.waitForCompletion(true);
/*
           Job job4 = new Job(conf, "Conferences");
           job4.setJarByClass(driver1.class);
           job4.setMapperClass(mapper4_1.class);
           job4.setReducerClass(reducer4_1.class);
           job4.setMapOutputKeyClass(Text.class);
           job4.setMapOutputValueClass(Text.class);
           job4.setOutputKeyClass(Text.class);
           job4.setOutputValueClass(Text.class);
           FileInputFormat.addInputPath(job4, new Path(args[0]));
           FileOutputFormat.setOutputPath(job4, new
Path(temp_path));
           job4.waitForCompletion(true);

           Job job5 = new Job(conf, "Conferences");
           job5.setJarByClass(driver1.class);
           job5.setMapperClass(mapper4_2.class);
           job5.setReducerClass(reducer4_2.class);
           job5.setMapOutputKeyClass(Text.class);
           job5.setMapOutputValueClass(IntWritable.class);
           job5.setOutputKeyClass(Text.class);
           job5.setOutputValueClass(IntWritable.class);
           FileInputFormat.addInputPath(job5, new Path(temp_path));
           FileOutputFormat.setOutputPath(job5, new
Path(args[1]+"/4"));
           return (job5.waitForCompletion(true) ? 0 : 1);*/
      }


      }
```

**Driver 2:**

```
package assign_2;

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

public class driver2 {
	@SuppressWarnings("deprecation")
	public static int main(String[] args) throws Exception {
		String temp_path = "temp";
		Configuration conf = new Configuration();
	Job job4 = new Job(conf, "Conferences");
	job4.setJarByClass(driver1.class);
	job4.setMapperClass(mapper4_1.class);
	job4.setReducerClass(reducer4_1.class);
	job4.setMapOutputKeyClass(Text.class);
	job4.setMapOutputValueClass(Text.class);
	job4.setOutputKeyClass(Text.class);
	job4.setOutputValueClass(Text.class);
	FileInputFormat.addInputPath(job4, new Path(args[0]));
	FileOutputFormat.setOutputPath(job4, new Path(temp_path));
	job4.waitForCompletion(true);

	Job job5 = new Job(conf, "Conferences");
	job5.setJarByClass(driver1.class);
	job5.setMapperClass(mapper4_2.class);
	job5.setReducerClass(reducer4_2.class);
	job5.setMapOutputKeyClass(Text.class);
	job5.setMapOutputValueClass(IntWritable.class);
	job5.setOutputKeyClass(Text.class);
	job5.setOutputValueClass(IntWritable.class);
	FileInputFormat.addInputPath(job5, new Path(temp_path));
	FileOutputFormat.setOutputPath(job5, new Path(args[1]+"/4"));
	return (job5.waitForCompletion(true) ? 0 : 1);
}
}
```

## Task I:

## Mapper:

```
package assign_2;

import java.io.IOException;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

public class mapper1 extends Mapper<LongWritable, Text, Text,
IntWritable> {

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

	{
		String str = value.toString();
		String[] words = str.split("\t");

		// for (String city : words){
          Text outputKey = new Text(words[3].trim());

          IntWritable outputValue = new IntWritable(1);

          con.write(outputKey, outputValue);
     }
     // }

}
```

## Reducer:

```
package assign_2;
import java.io.IOException;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

public class reducer1 extends Reducer<Text, IntWritable, Text,
IntWritable>

{

public void reduce(Text word, Iterable<IntWritable> values, Context
con) throws IOException, InterruptedException

{
     int count = 0;

        for(IntWritable value : values)

        {

        count += value.get();

        }

        con.write(word, new IntWritable(count));

     }

}
```

**Task 2:**

**Mapper:**

```
package assign_2;


import java.io.IOException;

import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

public class mapper2 extends Mapper<LongWritable, Text, Text, Text>
{

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

	{
		String str = value.toString();
		String[] words=str.split("\t");

		Text outputKey = new Text(words[3].trim());

		Text outputValue = new Text(words[2].trim());

		con.write(outputKey, outputValue);
		}
	}
```

**Reducer:**

```
package assign_2;

import java.io.IOException;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.Reducer.Context;

public class reducer2 extends Reducer<Text, Text, Text, Text>

{
	Text value = new Text();
	public void reduce(Text city, Iterable<Text> values, Context
con)	throws IOException, InterruptedException {
		StringBuilder b = new StringBuilder();
		for (Text conf : values) {
			b.append(conf.toString());
			b.append(", ");
		}
		value.set(b.toString());
		con.write(city, new Text(value));

	}
```

## Task 3:

## Mapper:

```
package assign_2;


import java.io.IOException;

import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

public class mapper3 extends Mapper<LongWritable, Text, Text, Text>
{

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

	{
		String str = value.toString();
		String[] words=str.split("\t");

		Text outputKey = new Text(words[0].trim());

		Text outputValue = new Text(words[3].trim());

		con.write(outputKey, outputValue);
		}

	}
```

## Reducer:

```
package assign_2;
import java.io.IOException;

import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

public class reducer3 extends Reducer<Text, Text, Text, Text>{
Text v=new Text();
public void reduce(Text conference, Iterable<Text> values, Context
con) throws IOException, InterruptedException {
	StringBuilder b=new StringBuilder();
	for(Text text:values){
		b.append(text.toString());
		b.append(", ");
	}
	b.setLength(b.length()-1);
	v.set(b.toString());
	con.write(conference, new Text(v));
	}
}
```

**Task 4:**
**Mapper I:**

```
package assign_2;


import java.io.IOException;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

public class mapper4_1 extends Mapper<Object, Text, Text, Text> {

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

	{

		String str = value.toString();
		String[] words=str.split("\t");
		String event = words[0].trim().concat(words[1].trim());

		Text city = new Text(words[3].trim());

		//IntWritable outputValue = new IntWritable(1);

		con.write(city, new Text(event.trim()));

	}
}
```

**Reducer I:**

```
package assign_2;

import java.io.IOException;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.Reducer.Context;

public class reducer4_1 extends Reducer<Text, Text, Text, Text>

{
	Text v = new Text();

	public void reduce(Text city, Iterable<Text> values, Context
con) throws IOException, InterruptedException {

		StringBuilder b = new StringBuilder();
		for (Text text : values) {
			b.append(text.toString());
			b.append(";");
		}
```

```
          v.set(b.toString().trim());

          con.write(city, new Text(v));

     }

}
```

**Mapper II:**

```
package assign_2;

import java.io.IOException;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

public class mapper4_2 extends Mapper<Object, Text, Text,
IntWritable> {
     public void map(Object key, Text value, Context con)
               throws IOException, InterruptedException {
     String str = value.toString();
     String input[] = str.split("\t");
     String city = input[0];
     String words[] = input[1].split(";");
     for (int i = 0; i < words.length; i++) {
          String y = words[i].substring(words[i].length() - 4,
words[i].length());
          Text outputKey = new Text(city.concat(y));
          con.write(outputKey, new IntWritable(1));
     }
}
}
```

**Reducer II:**

```
package assign_2;

import java.io.IOException;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

public class reducer4_2 extends Reducer<Text, IntWritable, Text,
IntWritable> {
     public void reduce(Text conference, Iterable<IntWritable>
values,
               Context con) throws IOException,
InterruptedException {

          int sum = 0;
```

```
            for (IntWritable value : values) {

                  sum += value.get();

            }

            con.write(conference, new IntWritable(sum));

      }

}
```