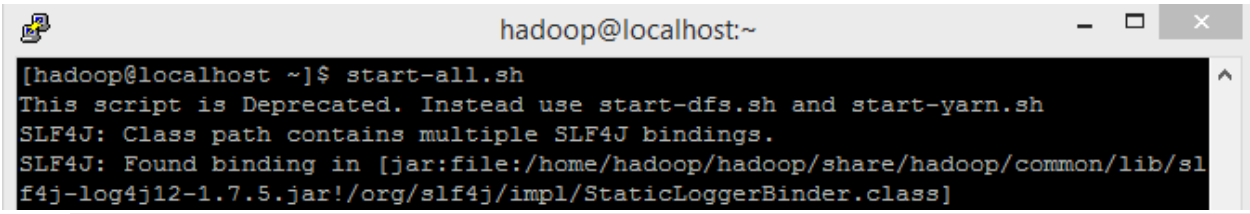
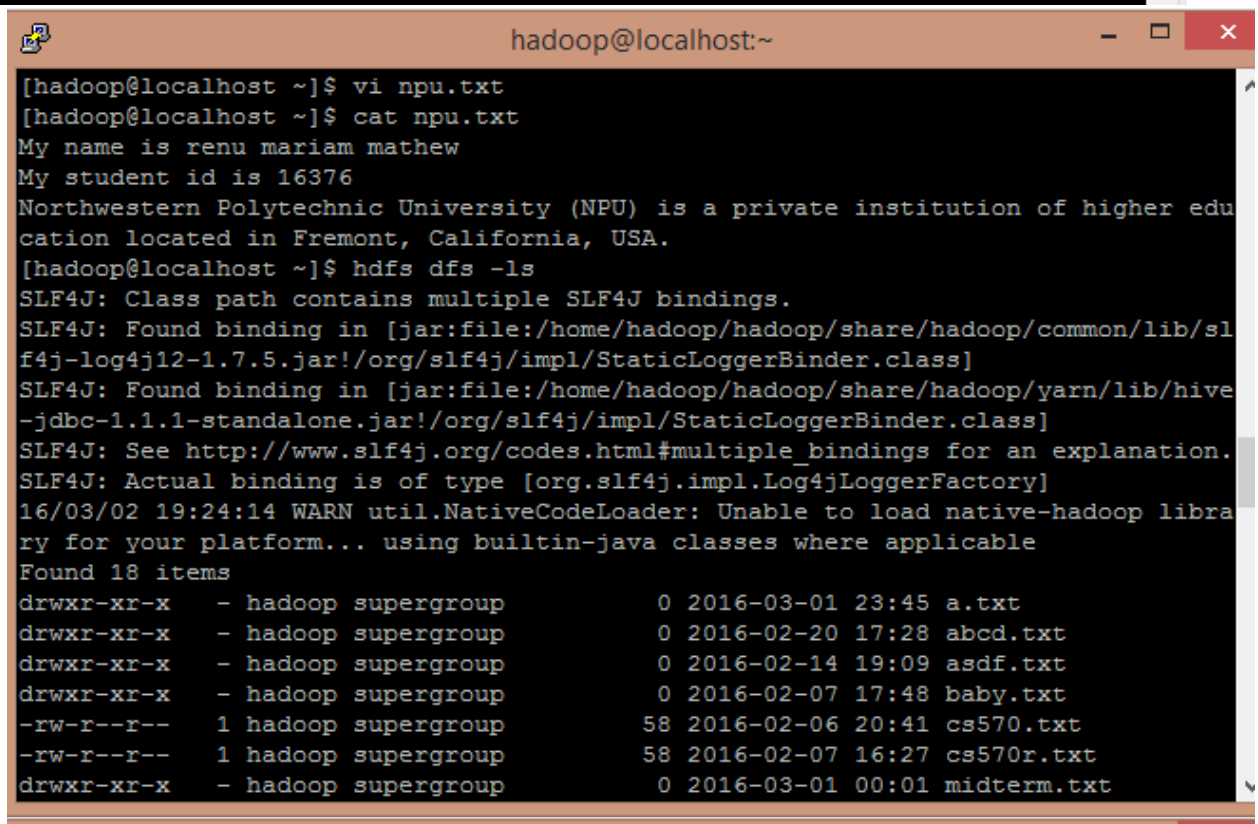
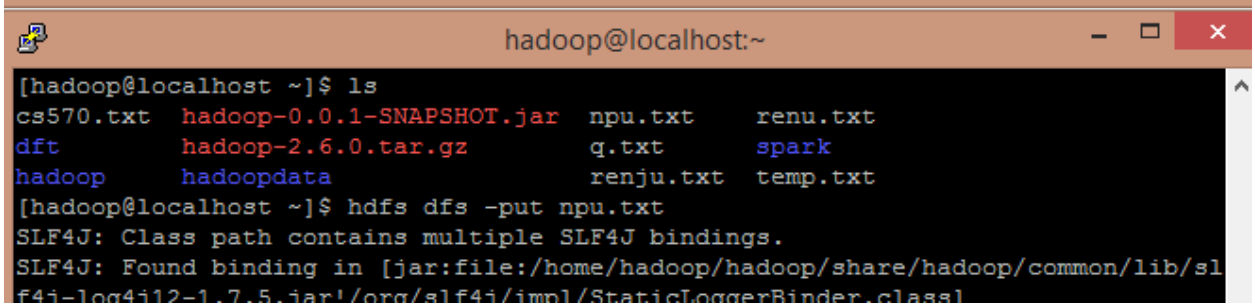
Name:Renu Mariam Mathew
Student Id:16376
Hw#2
Note: The first part of the document contains all the snapshots and the code for the program following that.

Initial steps:

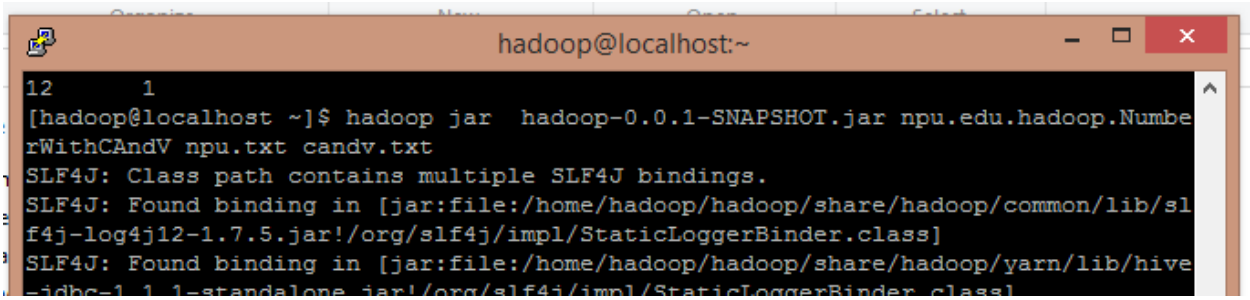


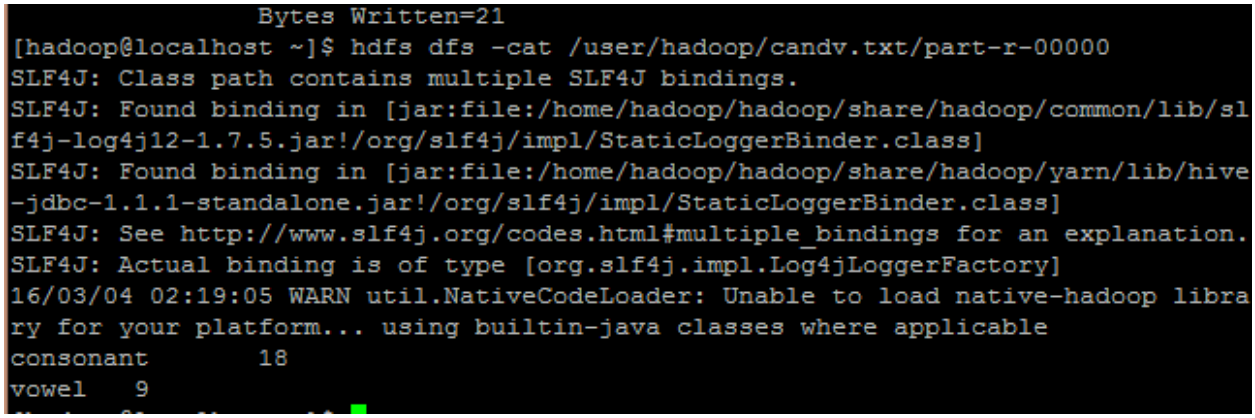




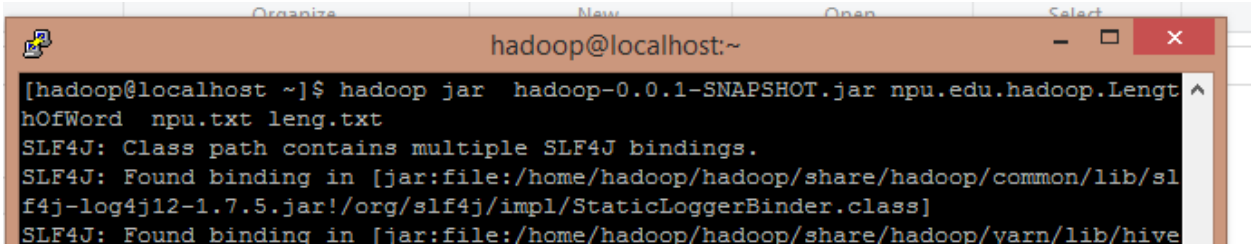
Name:Renu Mariam Mathew
Student Id:16376
Hw#2

1. Write a map reduce program to find the number of words that start with a consonant and vowel

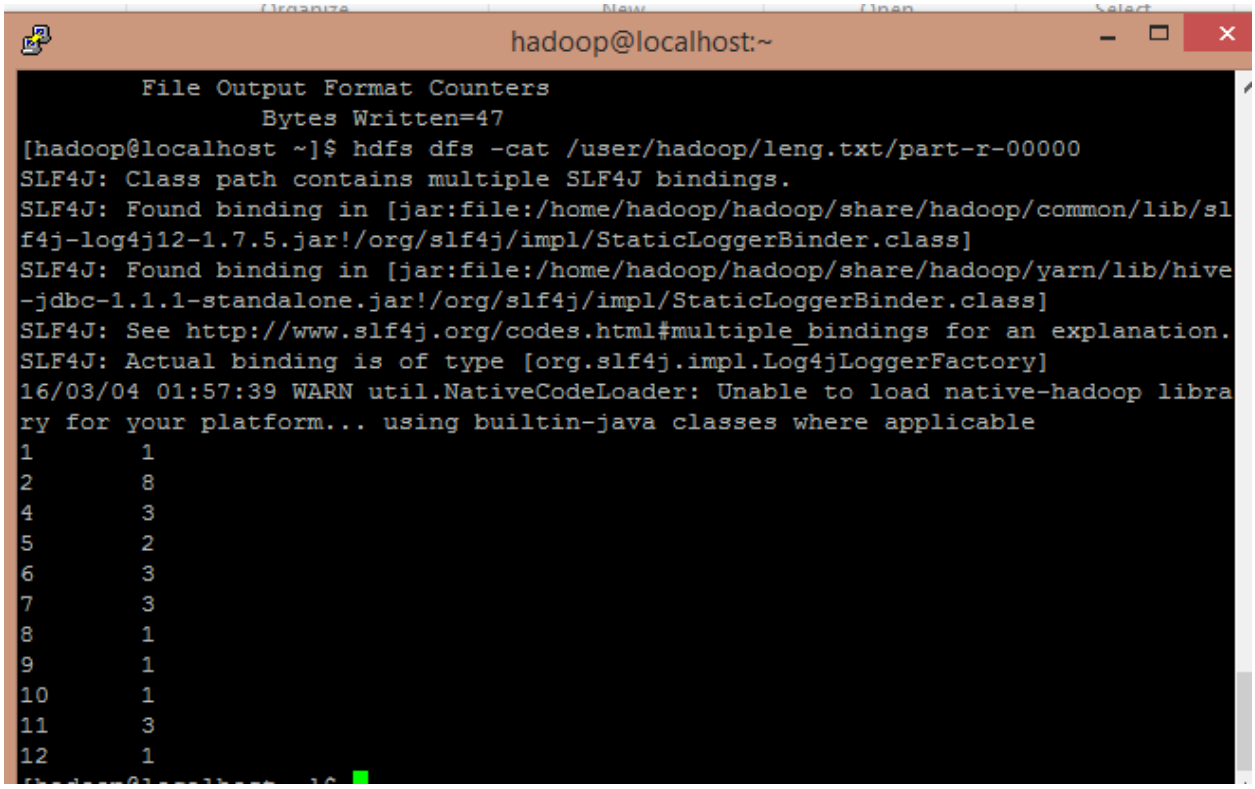




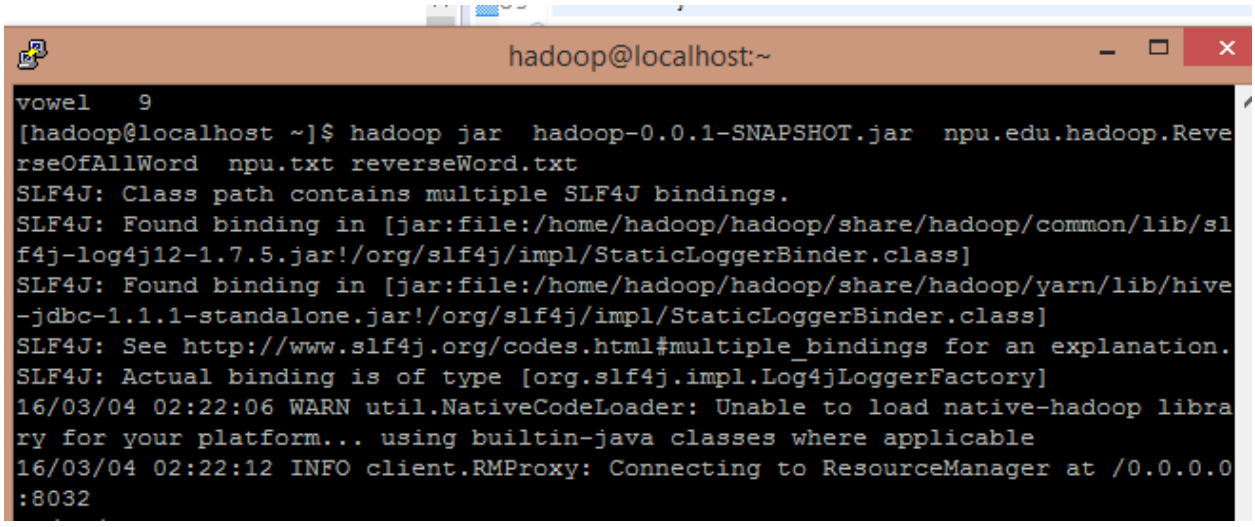
2. Write a map reduce program to find the length of each words in a document



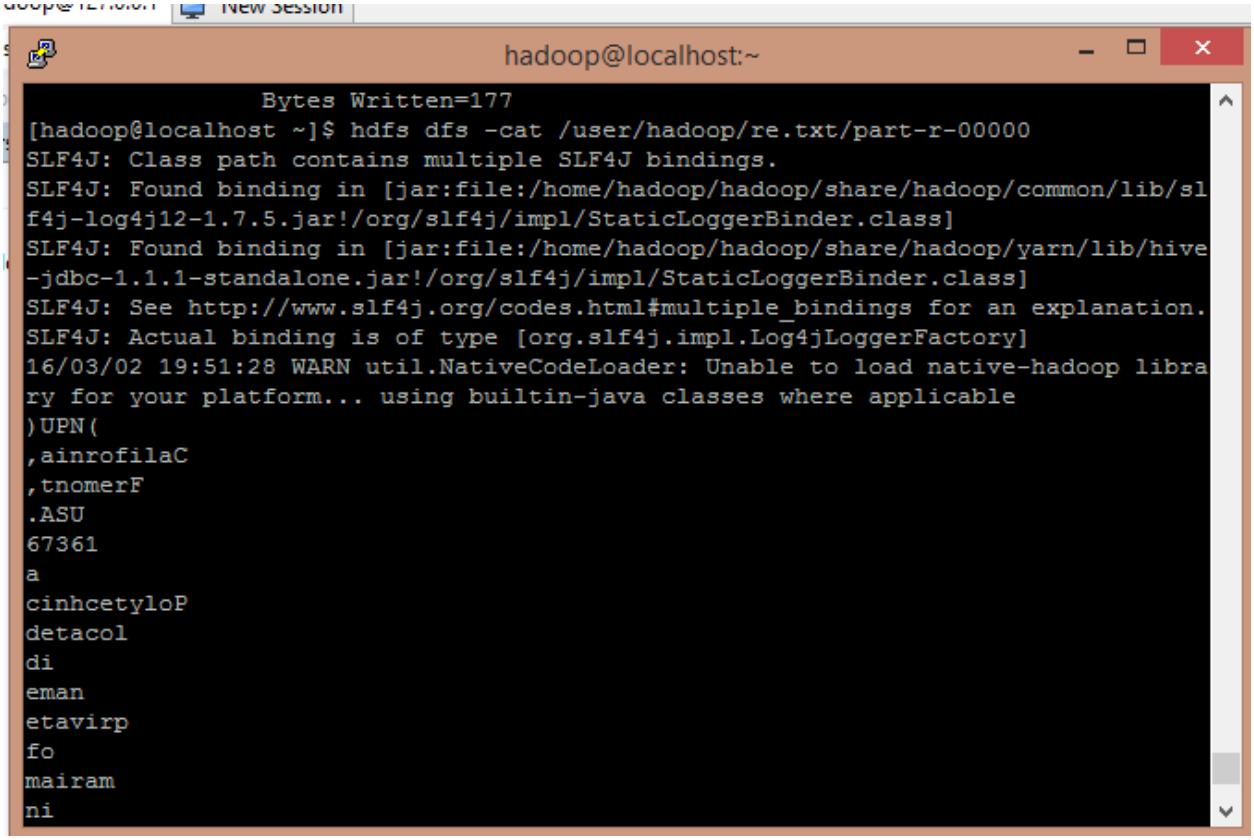
Name:Renu Mariam Mathew
Student Id:16376
Hw#2



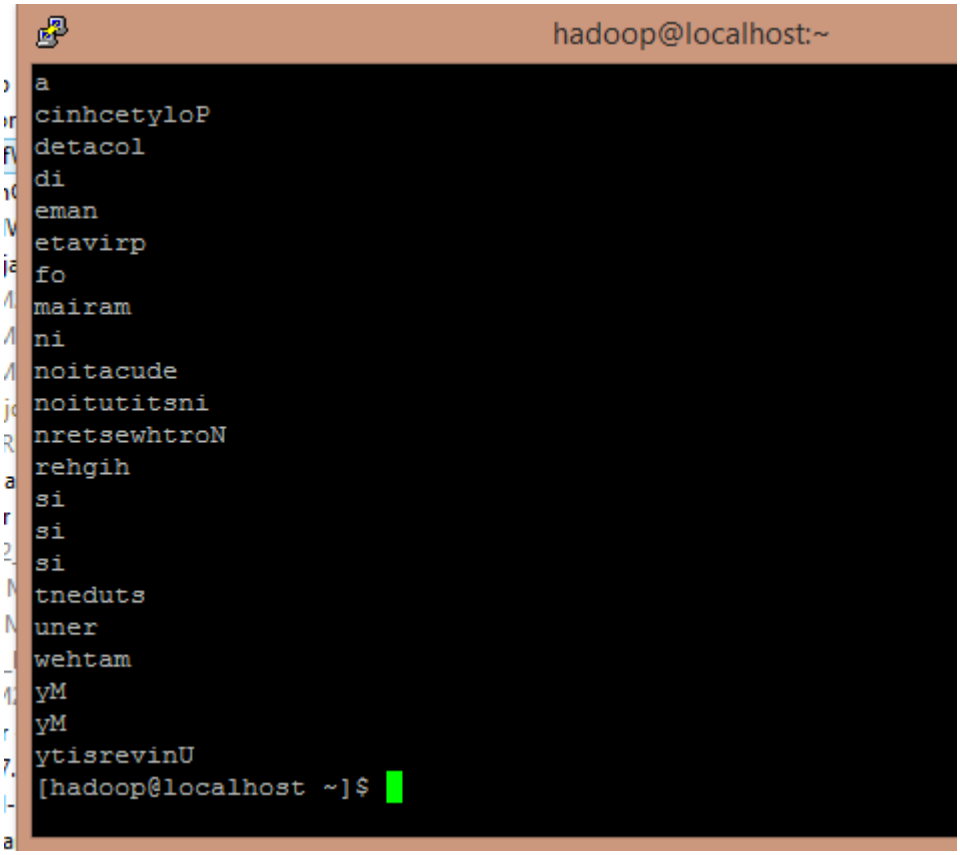
3. Write a map reduce program to reverse all the words in a document.



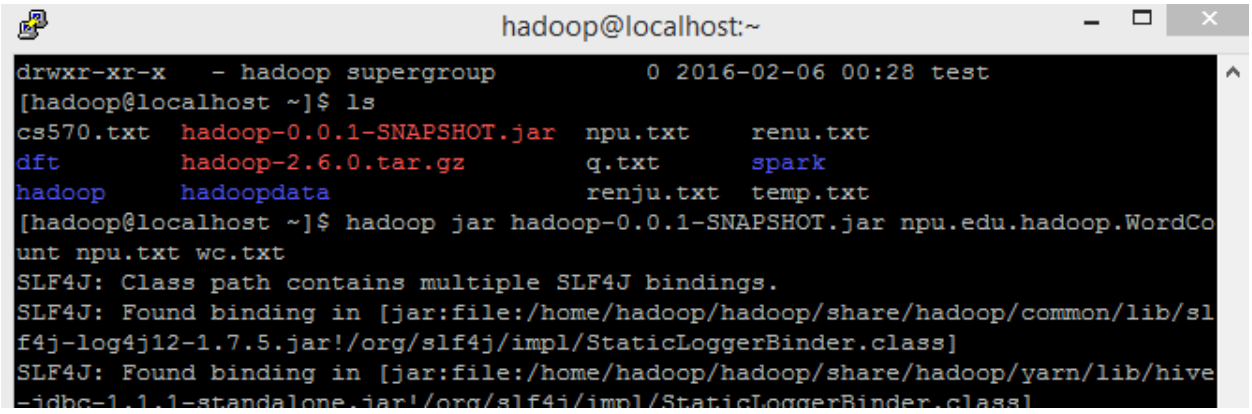
Name:Renu Mariam Mathew
Student Id:16376
Hw#2



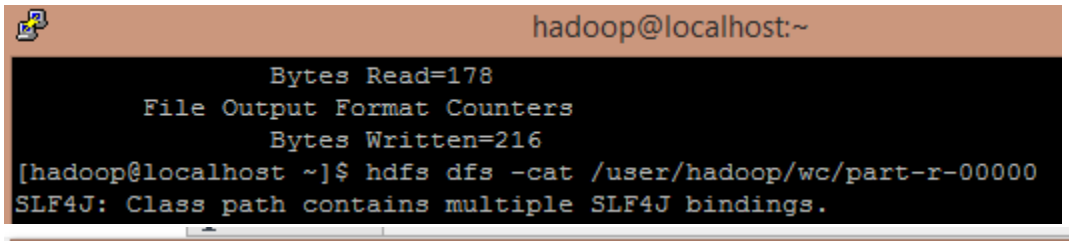
Name:Renu Mariam Mathew
Student Id:16376
Hw#2

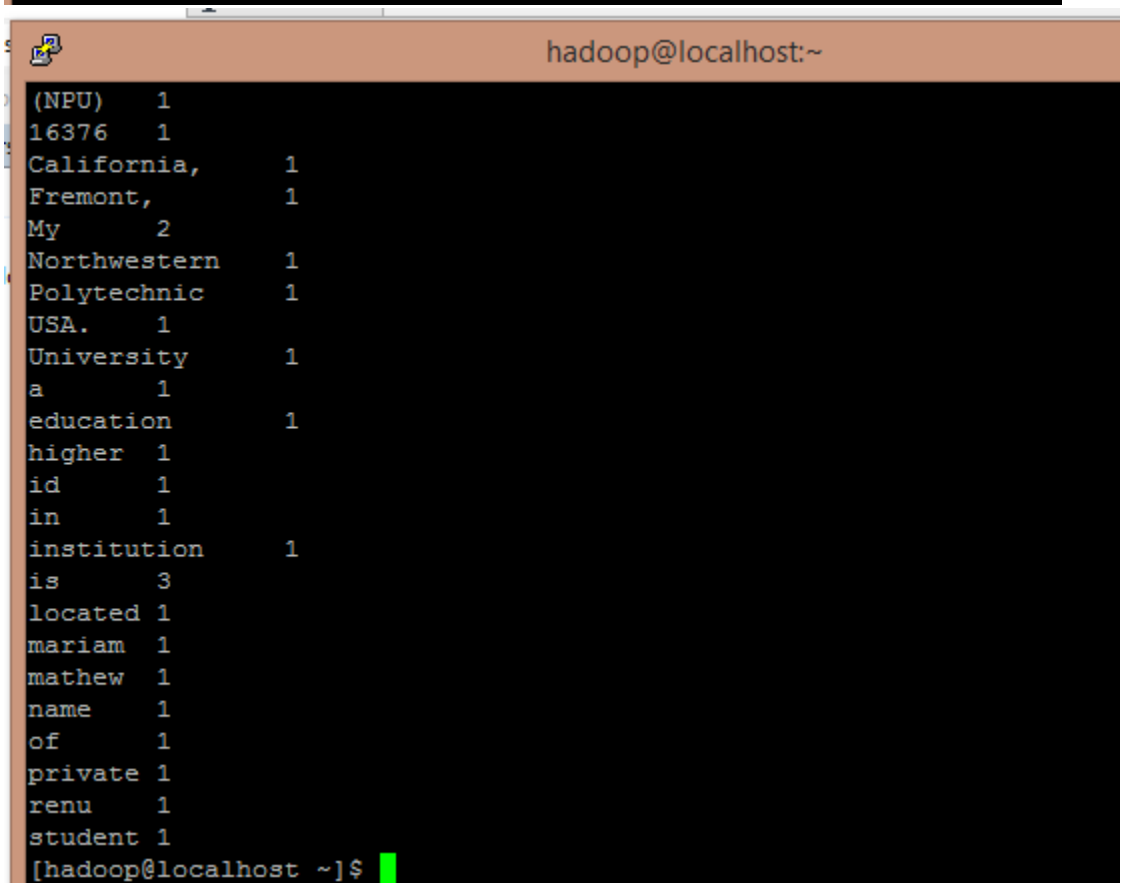


4. Write a map reduce program to count the total number of words in a document.



Name:Renu Mariam Mathew
Student Id:16376
Hw#2





Name:Renu Mariam Mathew
Student Id:16376
Hw#2

1.Write a map reduce program to find the number of words that start with a consonant and vowel.

```
package npu.edu.hadoop;

import java.io.IOException;
import java.util.StringTokenizer;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.Mapper.Context;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.util.GenericOptionsParser;

public class NumberWithCAndV {
      public static class TokenizerMapper extends Mapper<Object, Text, Text,
IntWritable> {
            private final static IntWritable one = new IntWritable(1);
            private Text word = new Text();

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

                  String a = "vowel";
                  String b = "consonant";

                  StringTokenizer itr = new
StringTokenizer(value.toString());
                  while (itr.hasMoreTokens()) {

                        String w = itr.nextToken();

                        if (w != null) {

                              if (w.toLowerCase().startsWith("a") ||
w.toLowerCase().startsWith("e")
```

Name:Renu Mariam Mathew
Student Id:16376
Hw#2

```
                                                || w.toLowerCase().startsWith("i")
|| w.toLowerCase().startsWith("o")
                                                || w.startsWith("u")) {

                                          word.set(a);

                                    } else {
                                          word.set(b);

                                    }

                              }
                              context.write(word, one);
                        }

                  }

            }

            public static class VowelsCountReducer extends Reducer<Text,
IntWritable, Text, IntWritable> {
                  private IntWritable result = new IntWritable();

                  public void reduce(Text key, Iterable<IntWritable> values,
Context context)
                              throws IOException, InterruptedException {
                        int sum = 0;
                        for (IntWritable value : values) {
                              sum += value.get();

                        }
                        result.set(sum);
                        context.write(key, result);

                  }
            }

            public static void main(String[] args) throws Exception {

                  Configuration conf = new Configuration();

                  String[] otherArgs = new GenericOptionsParser(conf,
args).getRemainingArgs();
                  if (otherArgs.length != 2) {
                        System.err.println("Usage:
NumberOfWordsWithConosantAndVowel <in> <out>");
                        System.exit(2);
                  }

                  Job job = new Job(conf, "NumberOfWordsWithConosantAndVowel");

                  job.setJarByClass(NumberWithCAndV.class);

                  job.setMapperClass(TokenizerMapper.class);
                  job.setCombinerClass(VowelsCountReducer.class);
```

Name:Renu Mariam Mathew
Student Id:16376
Hw#2

```
            job.setReducerClass(VowelsCountReducer.class);

            job.setOutputKeyClass(Text.class);
            job.setOutputValueClass(IntWritable.class);

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

      }
}
```

2.Write a map reduce program to find the length of each words in a document

```
package npu.edu.hadoop;

import java.io.IOException;
import java.util.StringTokenizer;
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.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.Reducer.Context;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.util.GenericOptionsParser;
import npu.edu.hadoop.WordCount.IntSumReducer;
import npu.edu.hadoop.WordCount.TokenizerMapper;
```

Name:Renu Mariam Mathew
Student Id:16376
Hw#2

```
public class LengthOfWord {
	public static class TokenizerMapper extends Mapper<Object, Text,
IntWritable, IntWritable> {

		private final static IntWritable one = new IntWritable(1);
		private IntWritable word = new IntWritable();

		public void map(Object key, Text value, Context context) throws
IOException, InterruptedException {
			StringTokenizer itr = new
StringTokenizer(value.toString());

			while (itr.hasMoreTokens()) {
				word.set(itr.nextToken().length());
				context.write(word, one);
			}

		}
	}

	public static class IntSumReducer extends Reducer<IntWritable,
IntWritable, IntWritable, IntWritable> {
		private IntWritable result = new IntWritable();

		public void reduce(IntWritable key, Iterable<IntWritable> values,
Context context)
				throws IOException, InterruptedException {
			int sum = 0;
			for (IntWritable val : values) {
				sum += val.get();
			}

			result.set(sum);
			context.write(key, result);
		}
	}

	public static void main(String[] args) throws Exception {
		Configuration conf = new Configuration();
		String[] otherArgs = new GenericOptionsParser(conf,
args).getRemainingArgs();
		if (otherArgs.length != 2) {
			System.err.println("Usage: LengthOfWord <in> <out>");
			System.exit(2);
		}
		Job job = new Job(conf, "LengthOfWord");
		job.setJarByClass(LengthOfWord.class);
		job.setMapperClass(TokenizerMapper.class);
		job.setCombinerClass(IntSumReducer.class);
		job.setReducerClass(IntSumReducer.class);
		job.setOutputKeyClass(IntWritable.class);
		job.setOutputValueClass(IntWritable.class);
		FileInputFormat.addInputPath(job, new Path(otherArgs[0]));
		FileOutputFormat.setOutputPath(job, new Path(otherArgs[1]));
		System.exit(job.waitForCompletion(true) ? 0 : 1);
```

Name:Renu Mariam Mathew
Student Id:16376
Hw#2

```
	}
}
```

3.Write a map reduce program to reverse all the words in a document.

```
package npu.edu.hadoop;

import java.io.IOException;
import java.util.StringTokenizer;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.NullWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;

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

public class ReverseOfAllWord {

	public static class ReverseMapper extends Mapper<LongWritable, Text,
Text, NullWritable> {
		private Text word = new Text();
		private final static IntWritable one = new IntWritable(1);

		public void map(LongWritable key, Text value, Context context)
throws IOException, InterruptedException {
			String line = value.toString().concat("\n");
			StringBuffer b = new StringBuffer(line);
			String result = (b.reverse().toString());
			StringTokenizer itr = new StringTokenizer(result);
			while (itr.hasMoreTokens()) {

				word.set(itr.nextToken());

				context.write(word, NullWritable.get());
			}
		}
	}

	public static void main(String[] args) throws Exception {

		Configuration conf = new Configuration();

		Job job = new Job(conf, "ReverseOfAllWord");
```

Name:Renu Mariam Mathew
Student Id:16376
Hw#2

```
            job.setJarByClass(ReverseOfAllWord.class);

            job.setMapperClass(ReverseMapper.class);
            // job.setCombinerClass(.class);

            // job.setReducerClass(.class);

            job.setOutputKeyClass(Text.class);
            job.setOutputValueClass(NullWritable.class);

            FileInputFormat.addInputPath(job, new Path(args[0]));
            FileOutputFormat.setOutputPath(job, new Path(args[1]));

            job.waitForCompletion(true);
      }
}
```

4.Write a map reduce program to count the total number of words in a document.

```
package npu.edu.hadoop;

import java.io.IOException;
import java.util.StringTokenizer;
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.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.util.GenericOptionsParser;

public class WordCount {
      public static class TokenizerMapper extends Mapper<Object, Text, Text,
IntWritable> {
            private final static IntWritable one = new IntWritable(1);
            private Text word = new Text();

            public void map(Object key, Text value, Context context) throws
IOException, InterruptedException {
                  StringTokenizer itr = new
StringTokenizer(value.toString());
                  while (itr.hasMoreTokens()) {
                        word.set(itr.nextToken());
                        context.write(word, one);
                  }
            }
```

Name:Renu Mariam Mathew
Student Id:16376
Hw#2

```
	}

	public static class IntSumReducer extends Reducer<Text, IntWritable,
Text, IntWritable> {
		private IntWritable result = new IntWritable();

		public void reduce(Text key, Iterable<IntWritable> values,
Context context)
				throws IOException, InterruptedException {
			int sum = 0;
			for (IntWritable val : values) {
				sum += val.get();
			}
			result.set(sum);
			context.write(key, result);
		}
	}

	public static void main(String[] args) throws Exception {
		Configuration conf = new Configuration();
		String[] otherArgs = new GenericOptionsParser(conf,
args).getRemainingArgs();
		if (otherArgs.length != 2) {
			System.err.println("Usage: WordCount <in> <out>");
			System.exit(2);
		}
		Job job = new Job(conf, "word count");
		job.setJarByClass(WordCount.class);
		job.setMapperClass(TokenizerMapper.class);
		job.setCombinerClass(IntSumReducer.class);
		job.setReducerClass(IntSumReducer.class);
		job.setOutputKeyClass(Text.class);
		job.setOutputValueClass(IntWritable.class);
		FileInputFormat.addInputPath(job, new Path(otherArgs[0]));
		FileOutputFormat.setOutputPath(job, new Path(otherArgs[1]));
		System.exit(job.waitForCompletion(true) ? 0 : 1);
	}
}
```