IMPLEMENT A WORD COUNT PROGRAM IN HADOOP.

Step 1: Set Up Hadoop (Single Node)

1. **Install Java:** java -version
2. **Download and Install Hadoop:**
   Get Hadoop from Apache Hadoop Official Website.
   Extract it to a directory (/usr/local/hadoop).
3. **Configure Hadoop:**
   Modify files: core-site.xml, hdfs-site.xml, mapred-site.xml, and yarn-site.xml.
4. **Format HDFS:** hdfs namenode -format
5. **Start Hadoop Services:** start-dfs.sh
   start-yarn.sh
6. Check the services are running and up : jps

Step 2: Set the MapRed classpath to the dfs :

Type the following command in terminal and hit enter.

mapred classpath.

Copy the output of above and paste in the below command.

export CLASSPATH="paste here class path output"

hit enter, classpath successfully set.

Step 3: Editing the mapreduce java program. (use gedit or nano)

1. Type the following command in terminal

gedit WordCount.java

2. copy the below java code and save the file :

//WordCount.java (Single File Version) //

```
import java.io.IOException;
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.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class WordCount {

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

      public void map(LongWritable key, Text value, Context context) throws IOException,
InterruptedException {
         String[] words = value.toString().split("\\s+");
         for (String w : words) {
            w = w.replaceAll("[^a-zA-Z]", "").toLowerCase();
            if (!w.isEmpty()) {
               word.set(w);
               context.write(word, one);
            }
         }
      }
   }

   public static class WordCountReducer extends Reducer<Text, IntWritable, Text, IntWritable> {
      public void reduce(Text key, Iterable<IntWritable> values, Context context) throws IOException,
InterruptedException {
         int sum = 0;
         for (IntWritable val : values) {
            sum += val.get();
         }
         context.write(key, new IntWritable(sum));
      }
```

```
    }

    public static void main(String[] args) throws Exception {
        if (args.length != 2) {
            System.err.println("Usage: WordCount <input path> <output path>");
            System.exit(-1);
        }

        Configuration conf = new Configuration();
        Job job = Job.getInstance(conf, "Word Count");

        job.setJarByClass(WordCount.class);
        job.setMapperClass(WordCountMapper.class);
        job.setReducerClass(WordCountReducer.class);

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

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

        System.exit(job.waitForCompletion(true) ? 0 : 1);
    }
}
```

Step 4: Compile and execute the code

Create the jar file

Prepare input data file

Run the mapreduce program

View the output file

- **A text input file (e.g., input.txt)**

Welcome to Hadoop session
Introduction to Hadoop
Introducing Hive
Hive session
Pig session

Word Count Logic in MapReduce

- **Mapper: Splits each line into words and emits (word, 1).**
- **Reducer: Sums all counts for each word**

REFER LINK : https://www.youtube.com/watch?v=6sK3LDY7Pp4

https://www.youtube.com/watch?v=UFxjn_y0K6I