HW Assignment 1 - CS267-Fall 2023

Hadoop/MapReduce Hands-on Exercises

Name : Sanket Kulkarni

Assignment Report Content:

Setup:

Steps	Errors Encountered
1. Instead of going with virtual box or vmware, I chose to dual boot my machine with ubuntu. I booted a usb with ubuntu OS and then installed ubuntu as well.	While installing the new booted OS, the boot enable flag of my BIOS was disabled, so I had to debug and enable this.
2. Installed JRE with version 11 and added the environment vars	N/A
3. Installed JDK	Added the path to env var so that it is accessible from hadoop env
4. Installed Hadoop and added its path to ./bashrc	N/A
5. I referred to the 7th point of setup on ubuntu given in the assignment. I tried setting up a multi node cluster as well, similar to one we have in production. While doing so, I changed Hadoop-env.xml, core-site.xml, hdfs-site.xml , yarn-site.xml	I did not place the right java home path initially, it was pointing to some other location. After debugging changed the file and jar file was able to run.
6. I created one hadoop user on my machine and now the directories of hadoop are completely isolated from the other code.	N/A
7. Installation of Python	N/A
8. Installation of Eclipse and setting up the project repo from where we will be exporting the jar files	Jar files needed in import of mapper and reducer were explicitly copied from hadoop commons to code repository.

Primary commands used in the setup:

```
# Download packages and install Java and SSH
$ sudo apt-get update
$ sudo apt-get install openjdk-11-jdk
$ sudo apt-get install openssh-server

# Update JAVA path in your bashrc file
$ sudo vi ~/.bashrc OR $ sudo gedit ~/.bashrc
  export JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64
$ source ~/.bashrc
```

Creating hadoop :

```
sudo adduser –ingroup hadoop hduser
```

Generate the SSH Key:

```
ssh-keygen –t rsa –P “”
```

enable SSH access:

```
sudo cat /home/hduser/.ssh/id_rsa.pub >> /home/hduser/.ssh/authorized_keys
```

$ sudo vi /usr/local/hadoop/conf/core-site.xml
To update below config files, using above command opening each file, one after the other

Open the mapred-site.xml using the vi editor -
$ sudo vi /usr/local/hadoop/conf/mapred-site.xml

To configure master:

```
$ sudo vi /usr/local/hadoop/conf/masters
```

To change the configs of HDFS-SITE.xml

```
sudo vi /usr/local/hadoop/conf/hdfs-site.xml
```

Setup: Config files screenshots

Conf/*-site.xml

```
<configuration>

<property>

<name>hadoop.tmp.dir</name>

<value>/home/snagpal/Desktop/hadoopdata</value>

<description></description>

</property>

<property> <

name>fs.defaultFS</name>

<value>hdfs://snagpal:54310</value>

<!-- <description>The Name of tegh default file system</description>-->

</property>

</configuration>
```

Conf/* mapred-site.xml

2. Conf/* mapred-site.xml

The next file to be configured is mapred-site.xml

```
<property>

<name>mapred.job.tracker</name>

<value>localhost:54311</value><!--localhost denotes the hostname-- > <description>The
host and port that the MapReduce job tracker runs at. If "local", then jobs are run in-
process as a single map and reduce task.

</description>

</property>
```

Conf/* hdfs-site.xml

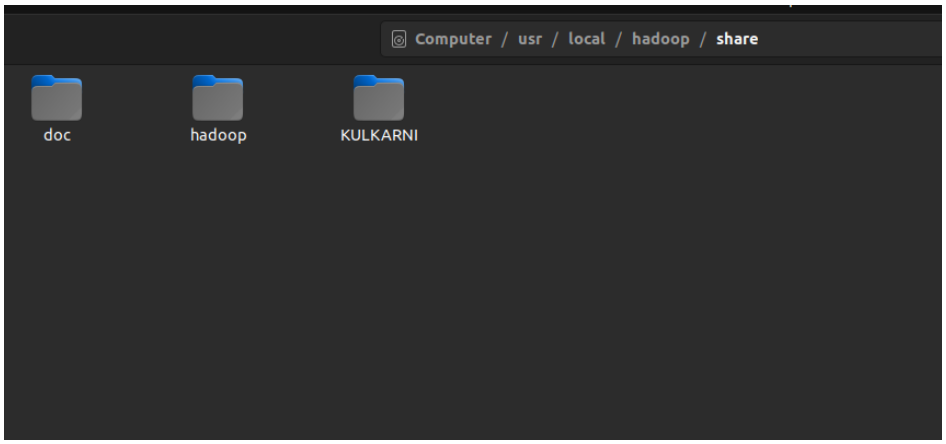
```
<property>

<name>dfs.replication</name>

<value>1</value>

<description>Default block replication. The actual number of replications can be
specified when the file is created. The default is used if replication is not specified
in create time.

</description>

</property>
```

Snapshots:

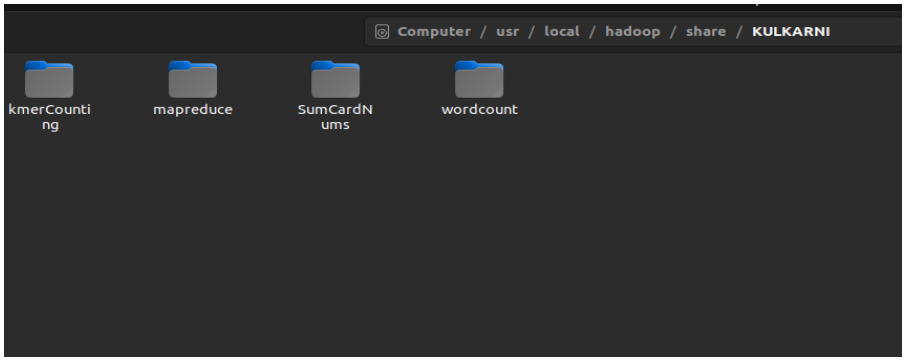
A Snapshot from .bashrc file in ubuntu, which showcases the Hadoop environment vars

```
export HADOOP_HOME=/usr/local/hadoop

export HADOOP_INSTALL=$HADOOP_HOME

export HADOOP_MAPRED_HOME=$HADOOP_HOME

export HADOOP_COMMON_HOME=$HADOOP_HOME

export HADOOP_HDFS_HOME=$HADOOP_HOME

export YARN_HOME=$HADOOP_HOME

export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native

export PATH=$PATH:$HADOOP_HOME/sbin:$HADOOP_HOME/bin

export HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib/native"
```

Folder Structure Snapshot:



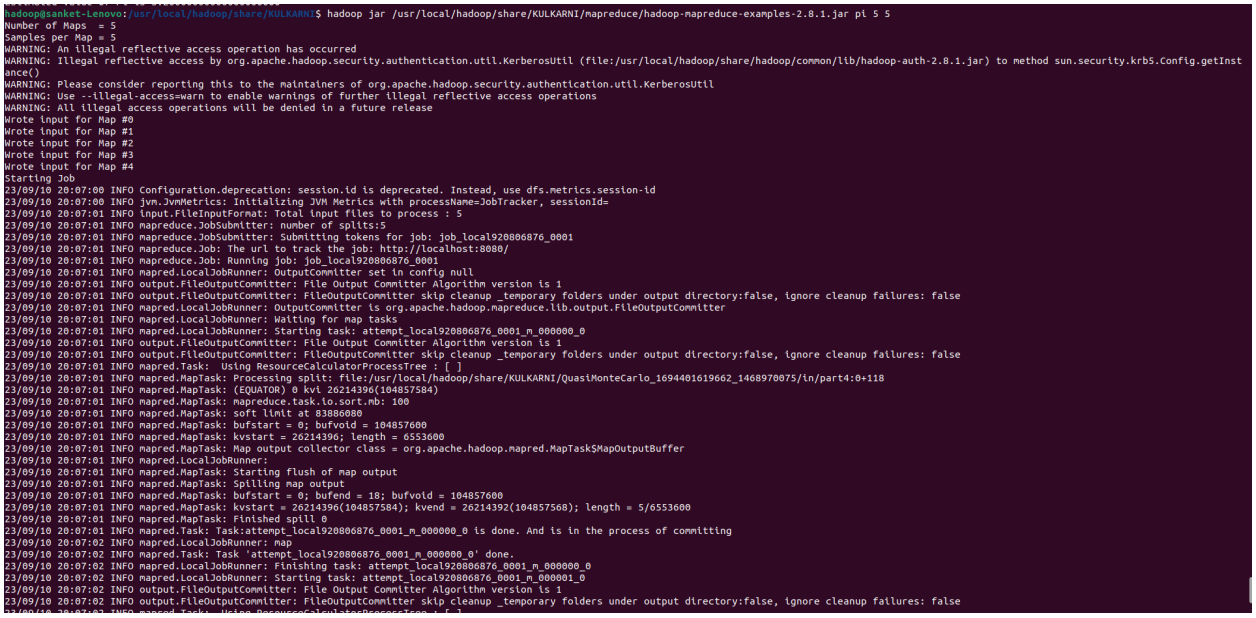
Snapshot of LASTNAME folder created with all code jars and files inside



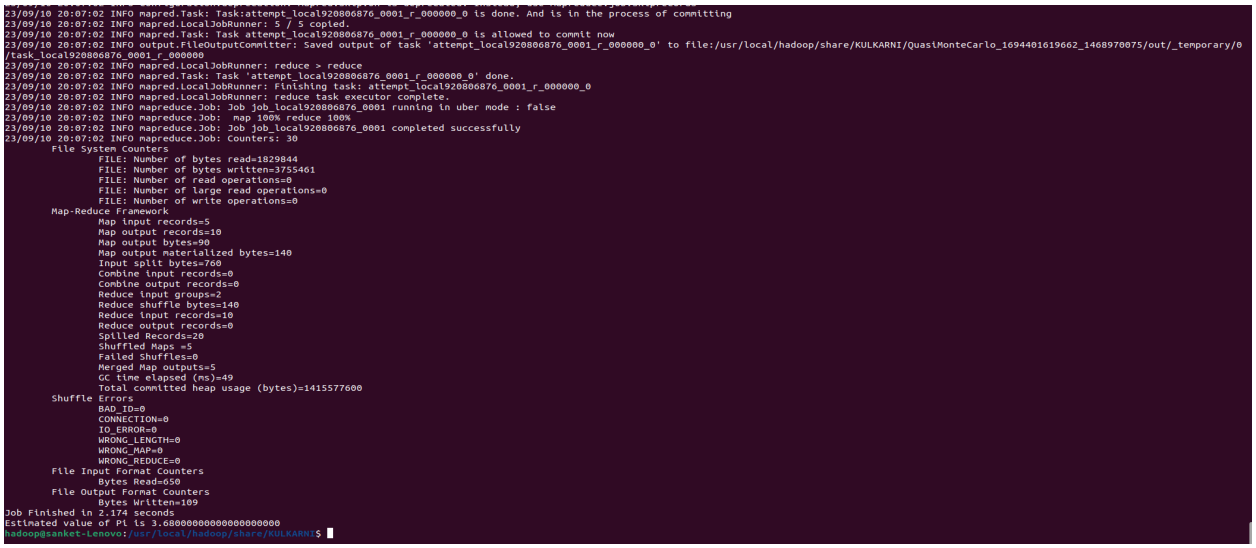
Exercise 1:

To run exercise one

- Created the KULKARNI (My last name) folder in the share folder inside hadoop.
- Copied mapreduce examples inside this folder
- Ran the pi value code using map reduce example.
- As asked, ran with 5 mapper and 5 reducer combination in the first go
- Achieved pi value as ~ 3.68



Output image for pi 5, 5:



As asked in the second task,

- Ran with 10 mapper and 10 reducer combination in the second go
- Achieved pi value as ~ 3.20

```
cp: target './KULKARNI/' is not a directory
hadoop@sanket-Lenovo:/usr/local/hadoop/share/hadoop$ cd ..
hadoop@sanket-Lenovo:/usr/local/hadoop/share$ cp -r hadoop/mapreduce ./KULKARNI/
hadoop@sanket-Lenovo:/usr/local/hadoop/share$ cd KULKARNI/
hadoop@sanket-Lenovo:/usr/local/hadoop/share/KULKARNI$ ls
mapreduce
hadoop@sanket-Lenovo:/usr/local/hadoop/share/KULKARNI$ hadoop jar /usr/local/hadoop/share/KULKARNI/mapreduce/hadoop-mapreduce-examples-2.8.1.jar pi 10 10
Number of Maps  = 10
Samples per Map = 10
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by org.apache.hadoop.security.authentication.util.KerberosUtil (file:/usr/local/hadoop/share/hadoop/common/lib/hadoop-auth-2.8.1.jar) to method sun.security.krb5.Config.getInst
ance()
WARNING: Please consider reporting this to the maintainers of org.apache.hadoop.security.authentication.util.KerberosUtil
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations
WARNING: All illegal access operations will be denied in a future release
Wrote input for Map #0
Wrote input for Map #1
Wrote input for Map #2
Wrote input for Map #3
Wrote input for Map #4
Wrote input for Map #5
Wrote input for Map #6
Wrote input for Map #7
Wrote input for Map #8
Wrote input for Map #9
Starting Job
23/09/10 19:16:47 INFO Configuration.deprecation: session.id is deprecated. Instead, use dfs.metrics.session-id
23/09/10 19:16:47 INFO jvm.JvmMetrics: Initializing JVM Metrics with processName=JobTracker, sessionId=
23/09/10 19:16:47 INFO input.FileInputFormat: Total input files to process : 10
23/09/10 19:16:47 INFO mapreduce.JobSubmitter: number of splits:10
23/09/10 19:16:48 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_local336743653_0001
23/09/10 19:16:48 INFO mapreduce.Job: The url to track the job: http://localhost:8080/
23/09/10 19:16:48 INFO mapreduce.Job: Running job: job_local336743653_0001
23/09/10 19:16:48 INFO mapred.LocalJobRunner: OutputCommitter set in config null
23/09/10 19:16:48 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 1
23/09/10 19:16:48 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders under output directory:false, ignore cleanup failures: false
23/09/10 19:16:48 INFO mapred.LocalJobRunner: OutputCommitter is org.apache.hadoop.mapreduce.lib.output.FileOutputCommitter
23/09/10 19:16:48 INFO mapred.LocalJobRunner: Waiting for map tasks
23/09/10 19:16:48 INFO mapred.LocalJobRunner: Starting task: attempt_local336743653_0001_m_000000_0
23/09/10 19:16:48 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 1
23/09/10 19:16:48 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders under output directory:false, ignore cleanup failures: false
23/09/10 19:16:48 INFO mapred.Task:  Using ResourceCalculatorProcessTree : [ ]
23/09/10 19:16:48 INFO mapred.MapTask: Processing split: file:/usr/local/hadoop/share/KULKARNI/QuasiMonteCarlo_1694398606054_468189241/in/part4:0+118
23/09/10 19:16:49 INFO mapred.MapTask: (EQUATOR) 0 kvi 26214396(104857584)
23/09/10 19:16:49 INFO mapred.MapTask: mapreduce.task.io.sort.mb: 100
```

Output Image for pi 10 10:

```
task_local336743653_0001_r_000000
23/09/10 19:16:51 INFO mapred.LocalJobRunner: reduce > reduce
23/09/10 19:16:51 INFO mapred.Task: Task 'attempt_local336743653_0001_r_000000_0' done.
23/09/10 19:16:51 INFO mapred.LocalJobRunner: Finishing task: attempt_local336743653_0001_r_000000_0
23/09/10 19:16:51 INFO mapred.LocalJobRunner: reduce task executor complete.
23/09/10 19:16:51 INFO mapreduce.Job:  map 100% reduce 100%
23/09/10 19:16:51 INFO mapreduce.Job: Job job_local336743653_0001 completed successfully
23/09/10 19:16:51 INFO mapreduce.Job: Counters: 30
        File System Counters
                FILE: Number of bytes read=3413926
                FILE: Number of bytes written=6902795
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
        Map-Reduce Framework
                Map input records=10
                Map output records=20
                Map output bytes=180
                Map output materialized bytes=280
                Input split bytes=1510
                Combine input records=0
                Combine output records=0
                Reduce input groups=2
                Reduce shuffle bytes=280
                Reduce input records=20
                Reduce output records=0
                Spilled Records=40
                Shuffled Maps =10
                Failed Shuffles=0
                Merged Map outputs=10
                GC time elapsed (ms)=77
                Total committed heap usage (bytes)=2595225600
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=1300
        File Output Format Counters
                Bytes Written=109
Job Finished in 3.954 seconds
Estimated value of Pi is 3.20000000000000000000
hadoop@sanket-Lenovo:/usr/local/hadoop/share/KULKARNI$
```

Question 1: What is the result of pi value of pi for each case? What is the difference?

Answer:

With the combination of 5 mapper and 5 reducer

- PI value : 3.68000000000000000000

With the combination of 10 mapper and 10 reducer

- PI Value: 3.20000000000000000000

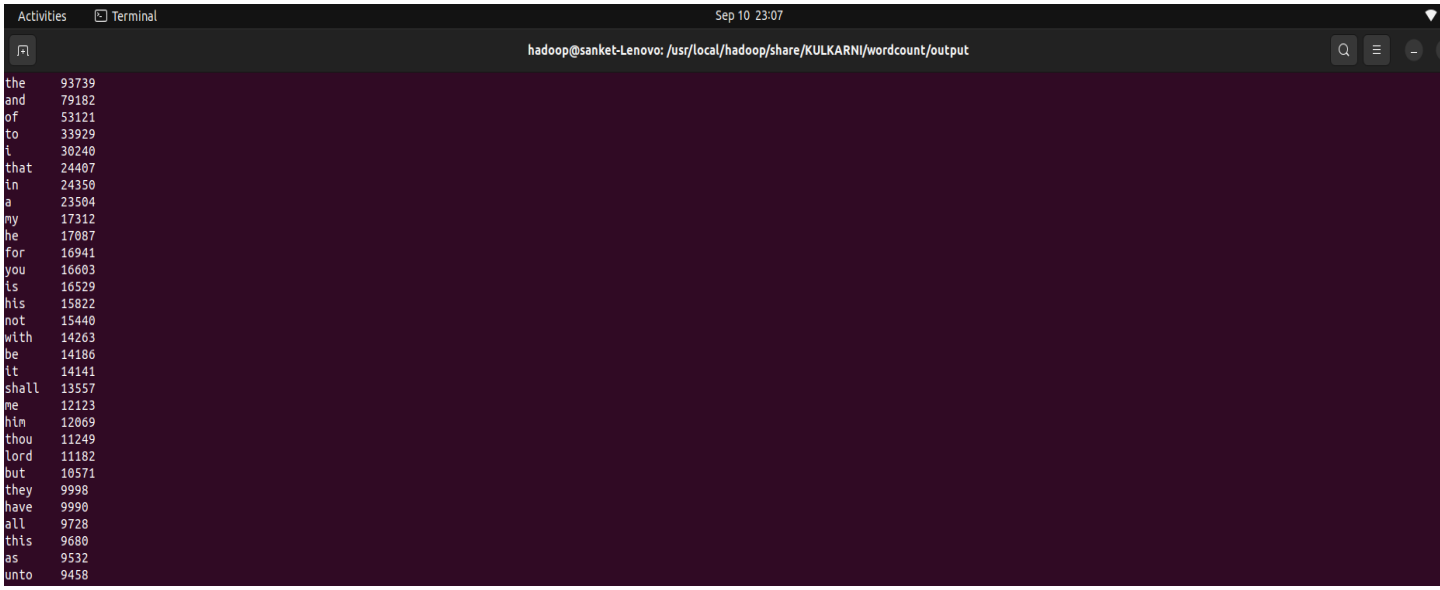
Observation: When we keep the mapper and reducer combination lower, the approx estimation is more error redundant. As the number of mappers and reducers increased, the PI value was estimated and then combined more accurately.

Exercise 2:

The next program to test is the Hadoop word count program. This example reads text files and counts how often words occur.

```
hadoop@sanket-Lenovo:/usr/local/hadoop/share/KULKARNI/wordcount/input$ hadoop jar /usr/local/hadoop/share/KULKARNI/mapreduce/hadoop-mapreduce-examples-2.8.1.jar wordcount /usr/local/hadoop/share/KULKARNI/wordcou
nt/input /usr/local/hadoop/share/KULKARNI/wordcount/output
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by org.apache.hadoop.security.authentication.util.KerberosUtil (file:/usr/local/hadoop/share/hadoop/common/lib/hadoop-auth-2.8.1.jar) to method sun.security.krb5.Config.getInst
ance()
WARNING: Please consider reporting this to the maintainers of org.apache.hadoop.security.authentication.util.KerberosUtil
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations
WARNING: All illegal access operations will be denied in a future release
23/09/10 22:07:58 INFO Configuration.deprecation: session.id is deprecated. Instead, use dfs.metrics.session-id
23/09/10 22:07:58 INFO jvm.JvmMetrics: Initializing JVM Metrics with processName=JobTracker, sessionId=
23/09/10 22:07:59 INFO input.FileInputFormat: Total input files to process : 1
23/09/10 22:07:59 INFO mapreduce.JobSubmitter: number of splits:1
23/09/10 22:07:59 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_local1808203099_0001
23/09/10 22:07:59 INFO mapreduce.Job: The url to track the job: http://localhost:8080/
23/09/10 22:07:59 INFO mapreduce.Job: Running job: job_local1808203099_0001
23/09/10 22:07:59 INFO mapred.LocalJobRunner: OutputCommitter set in config null
23/09/10 22:07:59 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 1
23/09/10 22:07:59 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders under output directory:false, ignore cleanup failures: false
23/09/10 22:07:59 INFO mapred.LocalJobRunner: OutputCommitter is org.apache.hadoop.mapreduce.lib.output.FileOutputCommitter
23/09/10 22:07:59 INFO mapred.LocalJobRunner: Waiting for map tasks
23/09/10 22:07:59 INFO mapred.LocalJobRunner: Starting task: attempt_local1808203099_0001_m_000000_0
23/09/10 22:07:59 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 1
23/09/10 22:07:59 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders under output directory:false, ignore cleanup failures: false
23/09/10 22:07:59 INFO mapred.Task:  Using ResourceCalculatorProcessTree : [ ]
23/09/10 22:07:59 INFO mapred.MapTask: Processing split: file:/usr/local/hadoop/share/KULKARNI/wordcount/input/bible+shakes.nopunc:0+9068074
23/09/10 22:07:59 INFO mapred.MapTask: (EQUATOR) 0 kvi 26214396(104857584)
23/09/10 22:07:59 INFO mapred.MapTask: mapreduce.task.io.sort.mb: 100
23/09/10 22:07:59 INFO mapred.MapTask: soft limit at 83886080
23/09/10 22:07:59 INFO mapred.MapTask: bufstart = 0; bufvoid = 104857600
23/09/10 22:07:59 INFO mapred.MapTask: kvstart = 26214396; length = 6553600
23/09/10 22:07:59 INFO mapred.MapTask: Map output collector class = org.apache.hadoop.mapred.MapTask$MapOutputBuffer
23/09/10 22:08:00 INFO mapreduce.Job: Job job_local1808203099_0001 running in uber mode : false
23/09/10 22:08:00 INFO mapreduce.Job:  map 0% reduce 0%
23/09/10 22:08:00 INFO mapred.LocalJobRunner:
23/09/10 22:08:00 INFO mapred.MapTask: Starting flush of map output
23/09/10 22:08:00 INFO mapred.MapTask: Spilling map output
23/09/10 22:08:00 INFO mapred.MapTask: bufstart = 0; bufend = 15919397; bufvoid = 104857600
23/09/10 22:08:00 INFO mapred.MapTask: kvstart = 26214396(104857584); kvend = 19277208(77108832); length = 6937189/6553600
23/09/10 22:08:02 INFO mapred.MapTask: Finished spill 0
23/09/10 22:08:02 INFO mapred.Task: Task:attempt_local1808203099_0001_m_000000_0 is done. And is in the process of committing
23/09/10 22:08:02 INFO mapred.LocalJobRunner: map
23/09/10 22:08:02 INFO mapred.Task: Task 'attempt_local1808203099_0001_m_000000_0' done.
23/09/10 22:08:02 INFO mapred.LocalJobRunner: Finishing task: attempt_local1808203099_0001_m_000000_0
```

```
                Reduce input records=41788
                Reduce output records=41788
                Spilled Records=83576
                Shuffled Maps =1
                Failed Shuffles=0
                Merged Map outputs=1
                GC time elapsed (ms)=76
                Total committed heap usage (bytes)=308281344
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=9068074
        File Output Format Counters
                Bytes Written=450684
hadoop@sanket-Lenovo:/usr/local/hadoop/share/KULKARNI/wordcount/input$ hadoop fs -get /usr/local/hadoop/share/KULKARNI/wordcount/output output
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by org.apache.hadoop.security.authentication.util.KerberosUtil (file:/usr/local/hadoop/share/hadoop/common/lib/hadoop-auth-2.8.1.jar) to method sun.security.krb5.Config.getInst
ance()
WARNING: Please consider reporting this to the maintainers of org.apache.hadoop.security.authentication.util.KerberosUtil
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations
WARNING: All illegal access operations will be denied in a future release
hadoop@sanket-Lenovo:/usr/local/hadoop/share/KULKARNI/wordcount/input$ cd
hadoop@sanket-Lenovo:~$ ls
auth_key  auth_key.pub  core-site.xml  hadoop-2.8.1.tar.gz  hadoop-env.sh  hdfs-site.xml  snap
hadoop@sanket-Lenovo:~$ cd /usr/local/hadoop/share/KULKARNI/wordcount/
hadoop@sanket-Lenovo:/usr/local/hadoop/share/KULKARNI/wordcount$ ls
input  output
hadoop@sanket-Lenovo:/usr/local/hadoop/share/KULKARNI/wordcount$ cd output/
hadoop@sanket-Lenovo:/usr/local/hadoop/share/KULKARNI/wordcount/output$
```



Understandings:

- Each mapper takes an input as a line and breaks it into an array of words. It then pushes the key, val pair of 1 as the count in the intermediate space.
- Reducer then picks these values and combines them. It stores the output in the output directory mentioned in the terminal path.

Command used to run:

```
hadoop jar
/usr/local/hadoop/share/KULKARNI/mapreduce/hadoop-mapreduce-examples-2.8.1.jar
wordcount /usr/local/hadoop/share/KULKARNI/wordcount/input
/usr/local/hadoop/share/KULKARNI/wordcount/output
```

Question 2: What are the top 10 most frequently used words in the corpus?

the	93739
and	79182
of	53121
to	33929
i	30240
that	24407
in	24350

a	23504
my	17312
he	17087

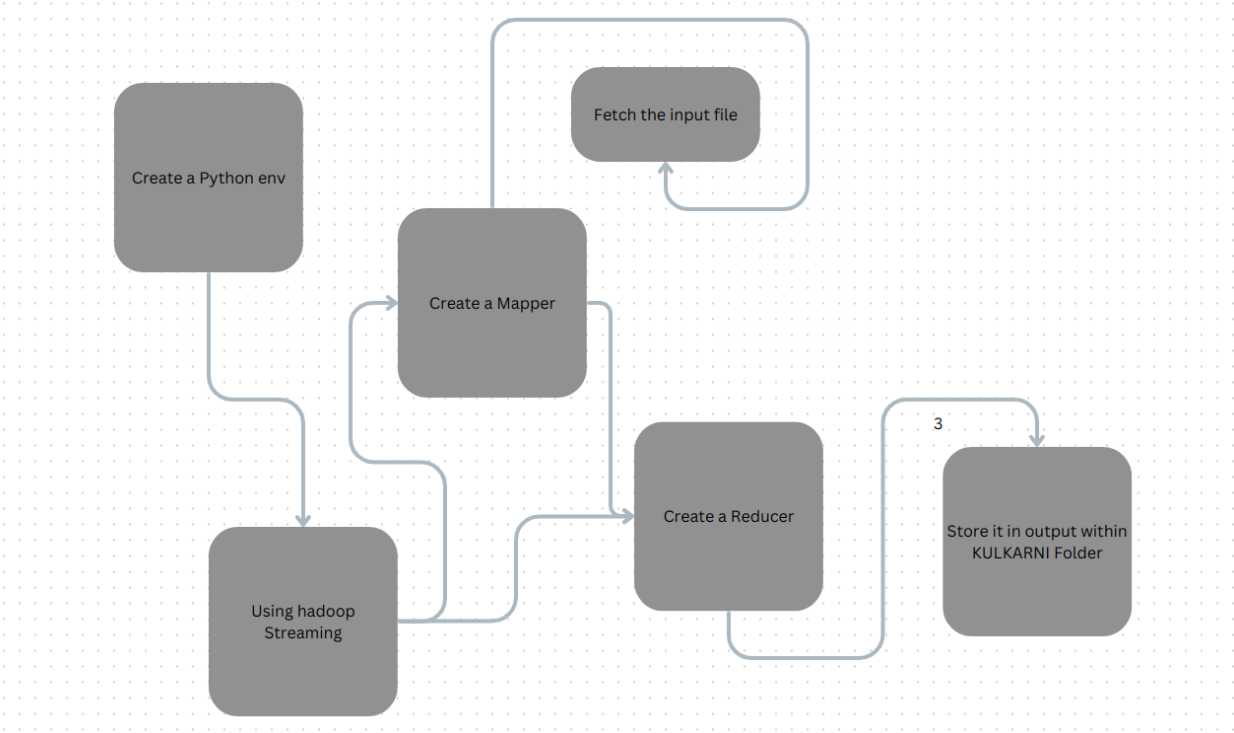
To get the most sorted one's in the file in descending order, I used below command of HDFS:

```
$ hadoop fs -cat part-000* | sort -k2,2nr > sorted_output_simple.txt
```

Exercise 3: Hadoop Kmer Counting

Approach:

- I have used python to write the mapper and reducer of this problem
- The purpose was to understand how hadoop streaming is used to run the mapper and reducers written in python



Sample Input 1: **ACACACAGT** to test whether the code is running properly or not. I stored this in a text file and then fed it as input to the reducer.

```
Name of the input file given : kmer_count_input.txt
Name of the output file generated : kmer_count_output
Time taken : few seconds
```

Command Used:

```
hadoop jar /usr/local/hadoop/share/hadoop/tools/lib/hadoop-streaming-*.jar \
-files kmer_count_mapper.py,kmer_count_reducer.py -mapper "kmer_count_mapper.py 9"
-reducer kmer_count_reducer.py \
-input kmer_count_input.txt -output kmer_count_output
```

```
hadoop@sanket-Lenovo:/usr/local/hadoop/share/KULKARNI/kmerCounting$ hadoop jar /usr/local/hadoop/share/hadoop/tools/lib/hadoop-streaming-*.jar -files kmer_count_mapper.py,kmer_count_reducer.
py -mapper "kmer_count_mapper.py 3" -reducer kmer_count_reducer.py -input kmer_count_input.txt -output kmer_count_output_simple
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by org.apache.hadoop.security.authentication.util.KerberosUtil (file:/usr/local/hadoop/share/hadoop/common/lib/hadoop-auth-2.8.1.jar) to method sun.securit
y.krb5.Config.getInstance()
WARNING: Please consider reporting this to the maintainers of org.apache.hadoop.security.authentication.util.KerberosUtil
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations
WARNING: All illegal access operations will be denied in a future release
23/09/20 01:39:03 INFO Configuration.deprecation: session.id is deprecated. Instead, use dfs.metrics.session-id
23/09/20 01:39:03 INFO jvm.JvmMetrics: Initializing JVM Metrics with processName=JobTracker, sessionId=
23/09/20 01:39:03 INFO jvm.JvmMetrics: Cannot initialize JVM Metrics with processName=JobTracker, sessionId= - already initialized
23/09/20 01:39:03 INFO mapred.FileInputFormat: Total input files to process : 1
23/09/20 01:39:03 INFO mapreduce.JobSubmitter: number of splits:1
23/09/20 01:39:03 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_local152041820_0001
23/09/20 01:39:03 INFO mapred.LocalDistributedCacheManager: Localized file:/usr/local/hadoop/share/KULKARNI/kmerCounting/kmer_count_mapper.py as file:/tmp/hadoop-hadoop/mapred/local/16951991
43745/kmer_count_mapper.py
23/09/20 01:39:03 INFO mapred.LocalDistributedCacheManager: Localized file:/usr/local/hadoop/share/KULKARNI/kmerCounting/kmer_count_reducer.py as file:/tmp/hadoop-hadoop/mapred/local/1695199
143746/kmer_count_reducer.py
23/09/20 01:39:04 INFO mapreduce.Job: The url to track the job: http://localhost:8080/
23/09/20 01:39:04 INFO mapreduce.Job: Running job: job_local152041820_0001
23/09/20 01:39:04 INFO mapred.LocalJobRunner: OutputCommitter set in config null
23/09/20 01:39:04 INFO mapred.LocalJobRunner: OutputCommitter is org.apache.hadoop.mapred.FileOutputCommitter
23/09/20 01:39:04 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 1
23/09/20 01:39:04 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders under output directory:false, ignore cleanup failures: false
23/09/20 01:39:04 INFO mapred.LocalJobRunner: Waiting for map tasks
23/09/20 01:39:04 INFO mapred.LocalJobRunner: Starting task: attempt_local152041820_0001_m_000000_0
23/09/20 01:39:04 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 1
23/09/20 01:39:04 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders under output directory:false, ignore cleanup failures: false
23/09/20 01:39:04 INFO mapred.Task:  Using ResourceCalculatorProcessTree : [ ]
23/09/20 01:39:04 INFO mapred.MapTask: Processing split: file:/usr/local/hadoop/share/KULKARNI/kmerCounting/kmer_count_input.txt:0+10
23/09/20 01:39:04 INFO mapred.MapTask: numReduceTasks: 1
23/09/20 01:39:04 INFO mapred.MapTask: (EQUATOR) 0 kvi 26214396(104857584)
23/09/20 01:39:04 INFO mapred.MapTask: mapreduce.task.io.sort.mb: 100
23/09/20 01:39:04 INFO mapred.MapTask: soft limit at 83886080
23/09/20 01:39:04 INFO mapred.MapTask: bufstart = 0; bufvoid = 104857600
23/09/20 01:39:04 INFO mapred.MapTask: kvstart = 26214396; length = 6553600
23/09/20 01:39:04 INFO mapred.MapTask: Map output collector class = org.apache.hadoop.mapred.MapTask$MapOutputBuffer
23/09/20 01:39:04 INFO streaming.PipeMapRed: PipeMapRed exec [/usr/local/hadoop/share/KULKARNI/kmerCounting/./kmer_count_mapper.py, 3]
23/09/20 01:39:04 INFO Configuration.deprecation: mapred.work.output.dir is deprecated. Instead, use mapreduce.task.output.dir
```

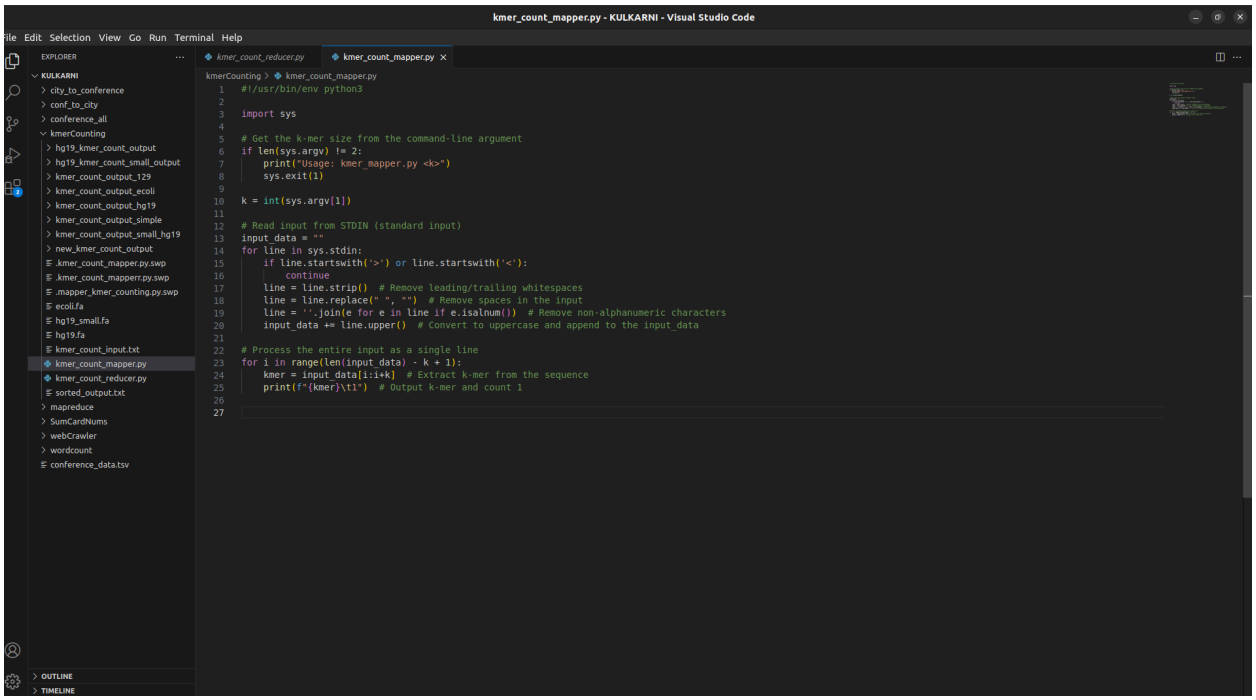
```
23/09/20 01:39:05 INFO mapreduce.Job:  map 100% reduce 100%
23/09/20 01:39:05 INFO mapreduce.Job: Job job_local152041820_0001 completed successfully
23/09/20 01:39:05 INFO mapreduce.Job: Counters: 30
        File System Counters
                FILE: Number of bytes read=270872
                FILE: Number of bytes written=925714
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
        Map-Reduce Framework
                Map input records=1
                Map output records=7
                Map output bytes=42
                Map output materialized bytes=62
                Input split bytes=123
                Combine input records=0
                Combine output records=0
                Reduce input groups=4
                Reduce shuffle bytes=62
                Reduce input records=7
                Reduce output records=4
                Spilled Records=14
                Shuffled Maps =1
                Failed Shuffles=0
                Merged Map outputs=1
                GC time elapsed (ms)=11
                Total committed heap usage (bytes)=471859200
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=10
        File Output Format Counters
                Bytes Written=36
23/09/20 01:39:05 INFO streaming.StreamJob: Output directory: kmer_count_output_simple
```

Output:

ACA	3
CAC	2
AGT	1
CAG	1

Question 3: What are the top 10 most frequently occurring 9-mers in E coli?

Code Snippet:



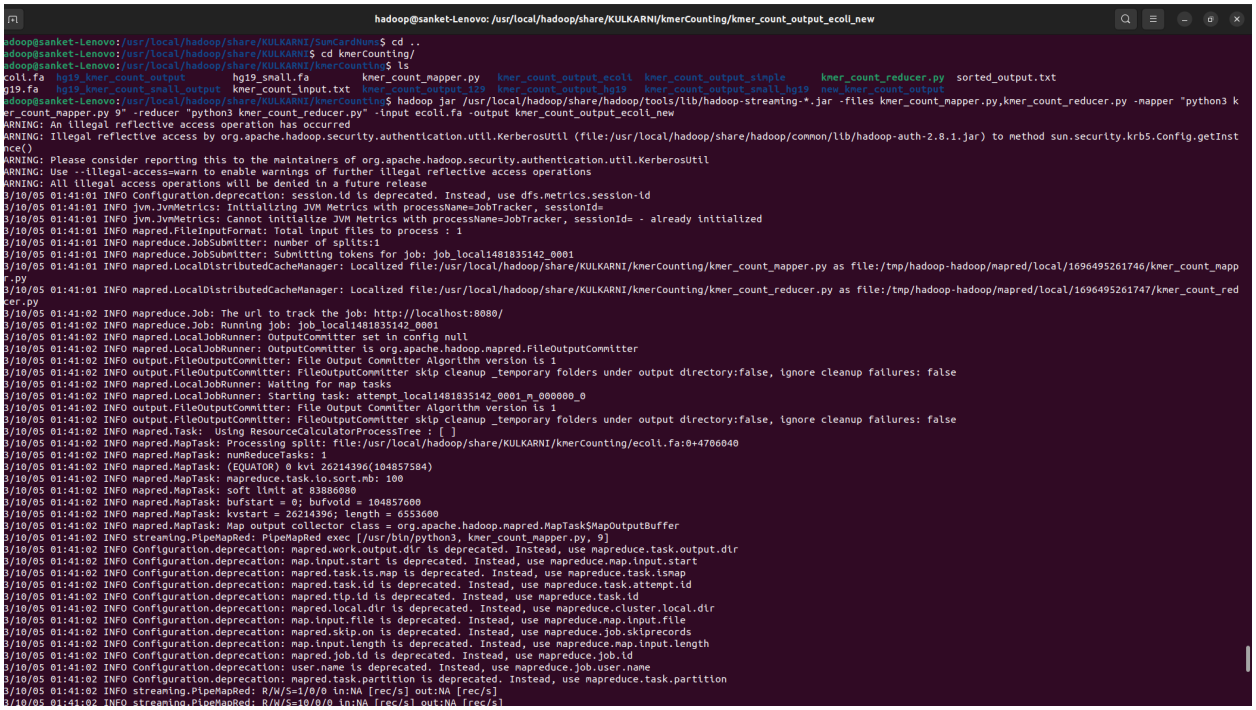
Changes done after comments:

- Added leading and trailing space check with a check for in between space
- Removed alphanumerics
- If line starts with '>' ignored that line as it is a genome
- Converted whole input to uppercase

Name of the input file given : ecoli.fa
Name of the output file generated : kmer_count_output_ecoli
Time taken : few minutes

Command Used:

```
hadoop jar /usr/local/hadoop/share/hadoop/tools/lib/hadoop-streaming-*.jar \
-files kmer_count_mapper.py,kmer_count_reducer.py -mapper "kmer_count_mapper.py 9"
-reducer kmer_count_reducer.py \
-input ecoli.fa -output kmer_count_output_ecoli
```



Output:

To get the most sorted one's in the file in descending order, I used below command of HDFS:

```
$ hadoop fs -cat part-000* | sort -k2,2nr > sorted_output_simple.txt
```

New output

Question 4: Run the above mapper and reducer with input file of hg-19

Name of the input file given : hg19.fa

Name of the output file generated : kmer_count_output_hg19

Time taken : about 4 hours

Command Used:

```
hadoop jar /usr/local/hadoop/share/hadoop/tools/lib/hadoop-streaming-*.jar \
-files kmer_count_mapper.py,kmer_count_reducer.py -mapper "kmer_count_mapper.py 9"
-reducer kmer_count_reducer.py \
-input hg19.fa -output kmer_count_output_hg19
```

Mapper Code Content:

1. Take the argument from command line regarding what should be the value of ‘k’ in the kmer
2. For every sentence in the file, remove whitespaces and trailing spaces. This will help us ensure text wrapping.
3. Create a subset of size k and then store it in intermediate space with count as 1
4. The storing will be done by adding tabs between the string and count

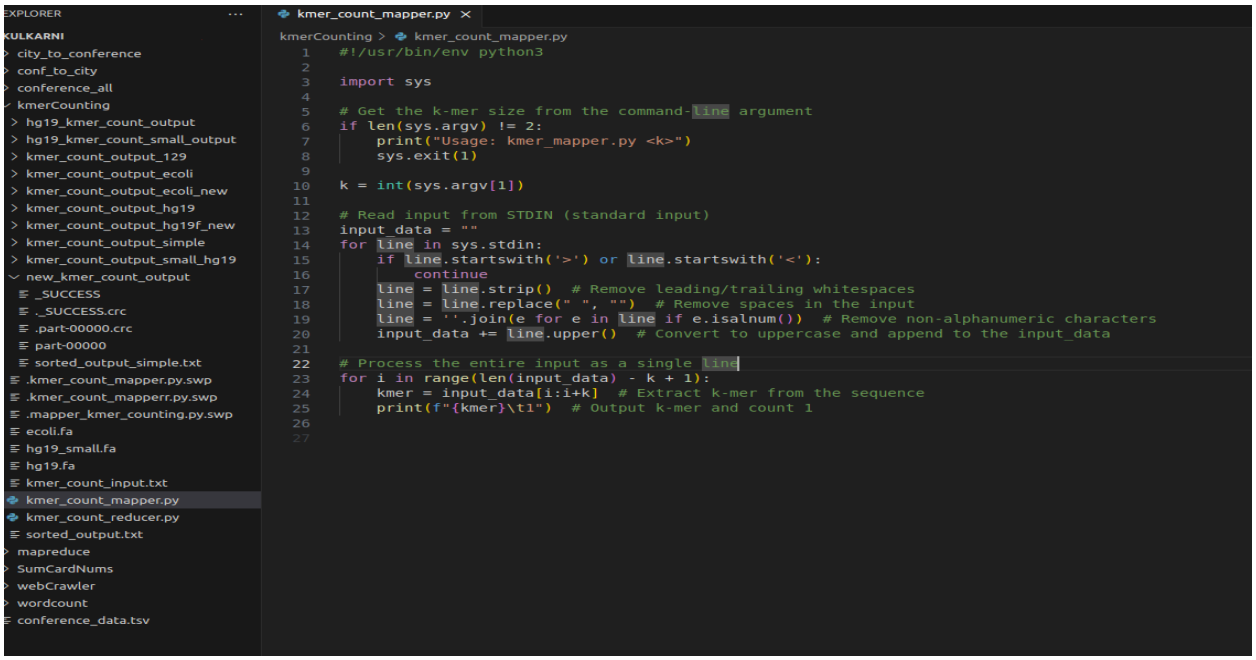
Reducer Code Content:

1. Split the input lines based on tabs that we added initially in the mapper
2. Convert the count which was stored earlier to integer value
3. If kmer already exist, increment the count
4. Else, add the kmer as a new entry

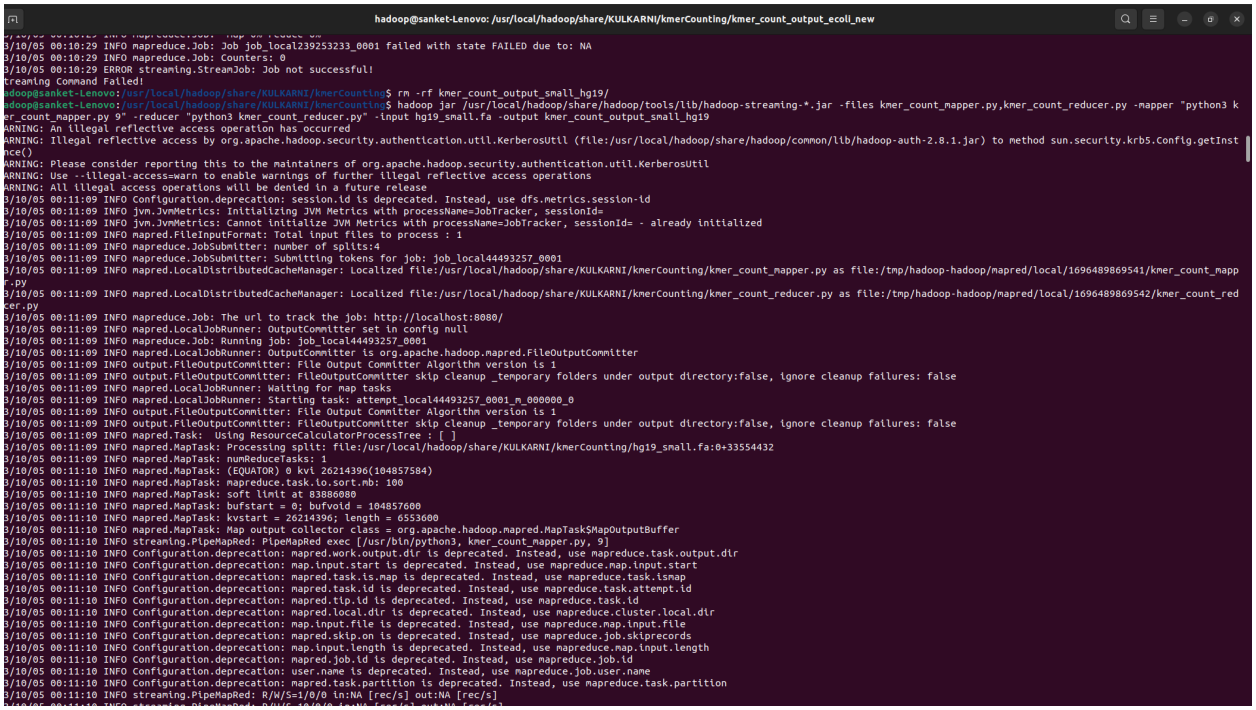
Changes done after comments:

- Added leading and trailing space check with a check for in between space
- Removed alphanumerics
- If line starts with ‘>’ ignored that line as it is a genome
- Converted whole input to uppercase

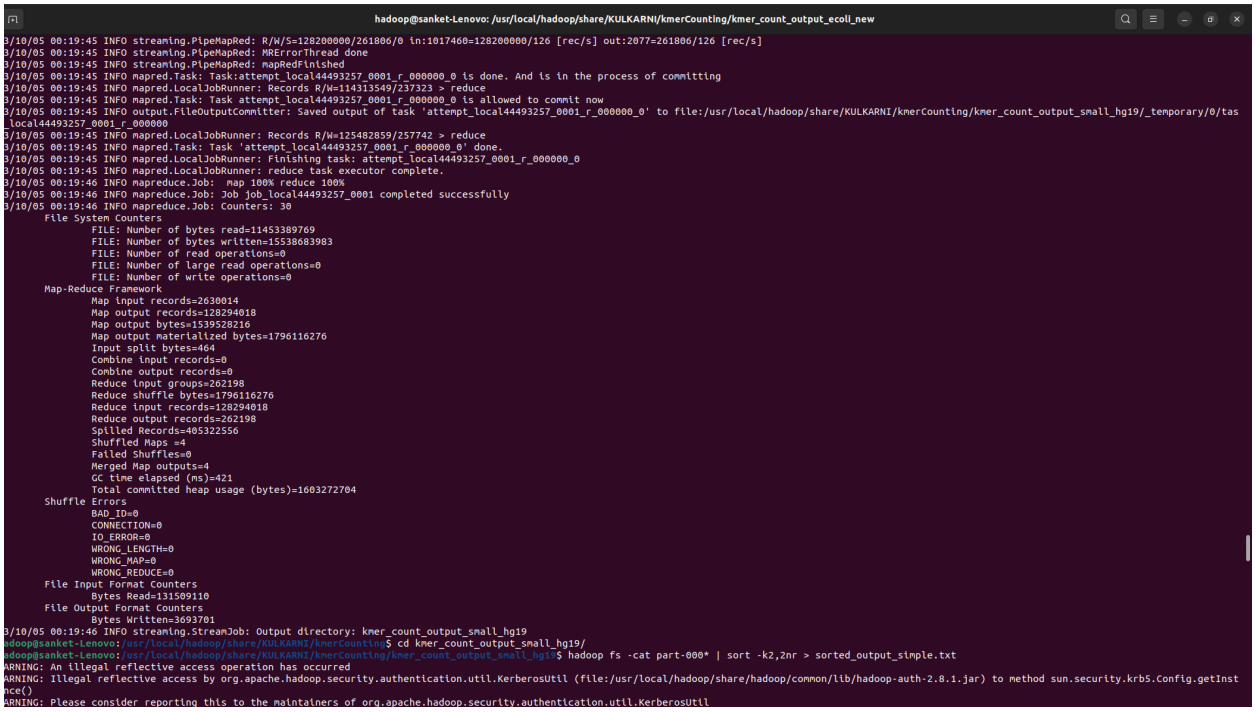
New Code Snippet:



Screenshot of hg-19 on Small Input:



Completion of hg-19 smaller

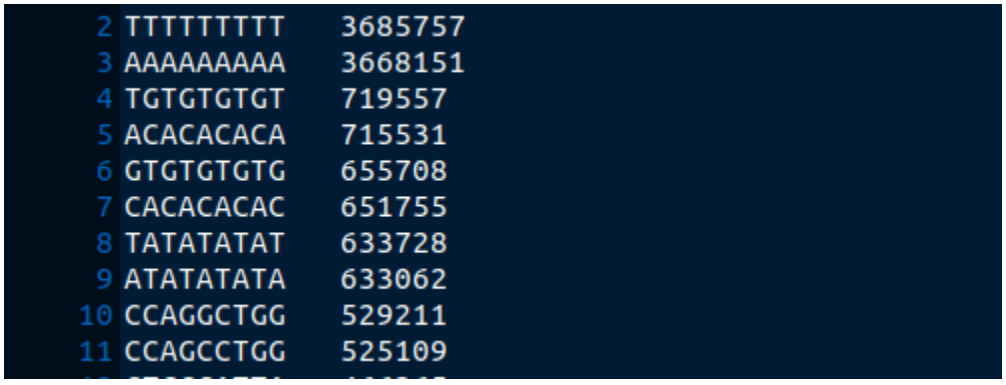


Output of hg-19 smaller:

Screenshot of hg-19

```
hadoop@sanket-Lenovo:/usr/local/hadoop/share/KULKARNI/kmerCounting$ hadoop jar /usr/local/hadoop/share/hadoop/tools/lib/hadoop-streaming-*.jar -files kmer_count_mapper.py,kmer_count_reducer.
py -mapper "kmer_count_mapper.py 9" -reducer kmer_count_reducer.py -input hg19.fa -output kmer_count_output_hg190
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by org.apache.hadoop.security.authentication.util.KerberosUtil (file:/usr/local/hadoop/share/hadoop/common/lib/hadoop-auth-2.8.1.jar) to method sun.securit
y.krb5.Config.getInstance()
WARNING: Please consider reporting this to the maintainers of org.apache.hadoop.security.authentication.util.KerberosUtil
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations
WARNING: All illegal access operations will be denied in a future release
23/09/20 01:30:06 INFO Configuration.deprecation: session.id is deprecated. Instead, use dfs.metrics.session-id
23/09/20 01:30:06 INFO jvm.JvmMetrics: Initializing JVM Metrics with processName=JobTracker, sessionId=
23/09/20 01:30:06 INFO jvm.JvmMetrics: Cannot initialize JVM Metrics with processName=JobTracker, sessionId= - already initialized
23/09/20 01:30:07 INFO mapred.FileInputFormat: Total input files to process : 1
23/09/20 01:30:07 INFO mapreduce.JobSubmitter: number of splits:96
23/09/20 01:30:07 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_local482311563_0001
23/09/20 01:30:07 INFO mapred.LocalDistributedCacheManager: Localized file:/usr/local/hadoop/share/KULKARNI/kmerCounting/kmer_count_mapper.py as file:/tmp/hadoop-hadoop/mapred/local/16951986
07643/kmer_count_mapper.py
23/09/20 01:30:07 INFO mapred.LocalDistributedCacheManager: Localized file:/usr/local/hadoop/share/KULKARNI/kmerCounting/kmer_count_reducer.py as file:/tmp/hadoop-hadoop/mapred/local/1695198
607644/kmer_count_reducer.py
23/09/20 01:30:07 INFO mapreduce.Job: The url to track the job: http://localhost:8080/
23/09/20 01:30:07 INFO mapreduce.Job: Running job: job_local482311563_0001
23/09/20 01:30:07 INFO mapred.LocalJobRunner: OutputCommitter set in config null
23/09/20 01:30:07 INFO mapred.LocalJobRunner: OutputCommitter is org.apache.hadoop.mapred.FileOutputCommitter
23/09/20 01:30:08 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 1
23/09/20 01:30:08 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders under output directory:false, ignore cleanup failures: false
23/09/20 01:30:08 INFO mapred.LocalJobRunner: Waiting for map tasks
23/09/20 01:30:08 INFO mapred.LocalJobRunner: Starting task: attempt_local482311563_0001_m_000000_0
23/09/20 01:30:08 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 1
23/09/20 01:30:08 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders under output directory:false, ignore cleanup failures: false
23/09/20 01:30:08 INFO mapred.Task:  Using ResourceCalculatorProcessTree : [ ]
23/09/20 01:30:08 INFO mapred.MapTask: Processing split: file:/usr/local/hadoop/share/KULKARNI/kmerCounting/hg19.fa:0+33554432
23/09/20 01:30:08 INFO mapred.MapTask: numReduceTasks: 1
23/09/20 01:30:08 INFO mapred.MapTask: (EQUATOR) 0 kvi 26214396(104857584)
23/09/20 01:30:08 INFO mapred.MapTask: mapreduce.task.io.sort.mb: 100
23/09/20 01:30:08 INFO mapred.MapTask: soft limit at 83886080
23/09/20 01:30:08 INFO mapred.MapTask: bufstart = 0; bufvoid = 104857600
23/09/20 01:30:08 INFO mapred.MapTask: kvstart = 26214396; length = 6553600
23/09/20 01:30:08 INFO mapred.MapTask: Map output collector class = org.apache.hadoop.mapred.MapTask$MapOutputBuffer
23/09/20 01:30:08 INFO streaming.PipeMapRed: PipeMapRed exec [/usr/local/hadoop/share/KULKARNI/kmerCounting/./kmer_count_mapper.py, 9]
23/09/20 01:30:08 INFO Configuration.deprecation: mapred.work.output.dir is deprecated. Instead, use mapreduce.task.output.dir
23/09/20 01:30:08 INFO Configuration.deprecation: map.input.start is deprecated. Instead, use mapreduce.map.input.start
```

New Output for hg19:

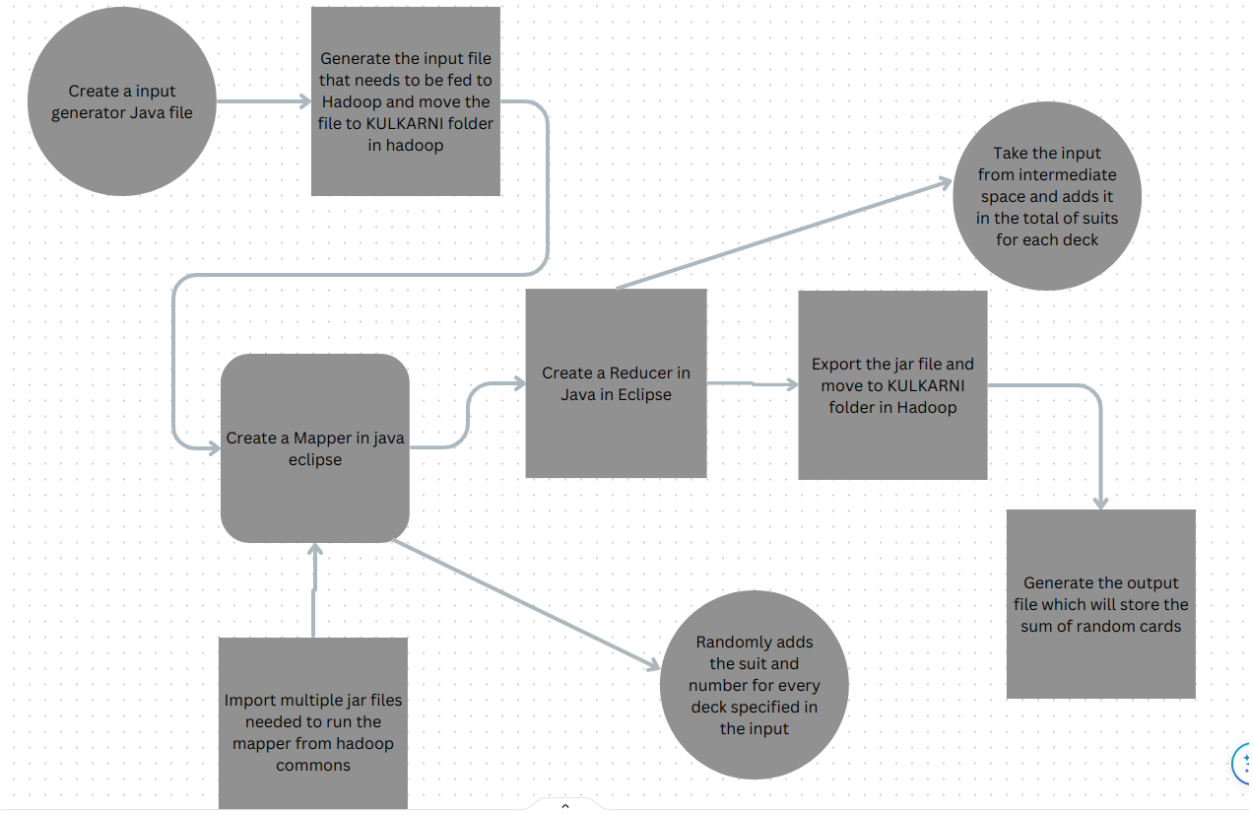


Total file looks like :

```
TTTTTTTTT   3685757
AAAAAAAAA   3668151
TGTGTGTGT   719557
ACACACACA   715531
GTGTGTGTG   655708
CACACACAC   651755
TATATATAT   633728
ATATATATA   633062
CCAGGCTGG   529211
CCAGCCTGG   525109
CTGGGATTA   466365
TAATCCCAG   465841
GATTACAGG   464772
CCTGTAATC   464113
GGATTACAG   464002
CTGTAATCC   463017
CTCAGCCTC   461167
GAGGCTGAG   459817
GCTGGGATT   455382
AATCCCAGC   454950
GGGATTACA   454308
TGTAATCCC   453943
TCAGCCTCC   451096
GGAGGCTGA   450156
GTAATCCCA   449030
TGGGATTAC   449014
CCTCAGCCT   446611
AGGCTGAGG   444230
GCCTCCCAA   429840
TTGGGAGGC   427616
CTTTTTTTT   403102
AAAAAAAAG   399876
TTTTATTTT   395116
AAAATAAAA   392797
TTTTTTTTG   384581
CAAAAAAAA   382277
CCCAGGCTG   375397
TTTTCTTTT   375307
ATTTTTTTT   374903
TTTGTATTT   374715
AAATACAAA   373791
AAAAAAAAT   373645
AAAAGAAAA   372483
CAGCCTGGG   372483
TTTCTTTTT   365231
TGTATTTTT   365183
```

Exercise 4: Hadoop Playing Cards Counting

Approach:

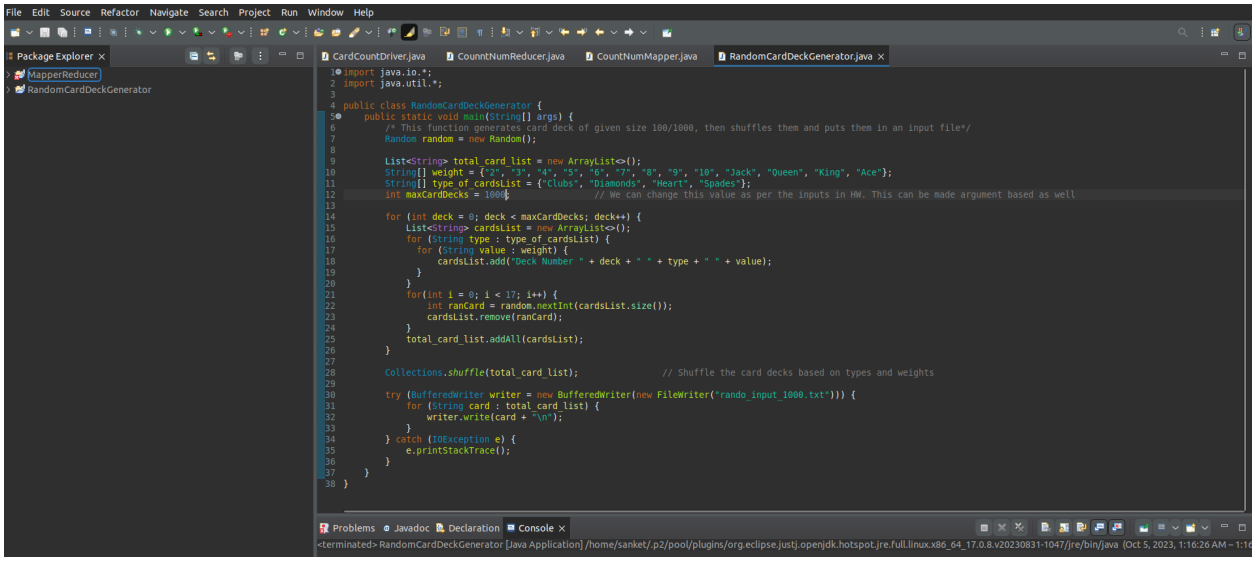


Source Code Snips:

Generation of cards randomly: Considered for face card as well now

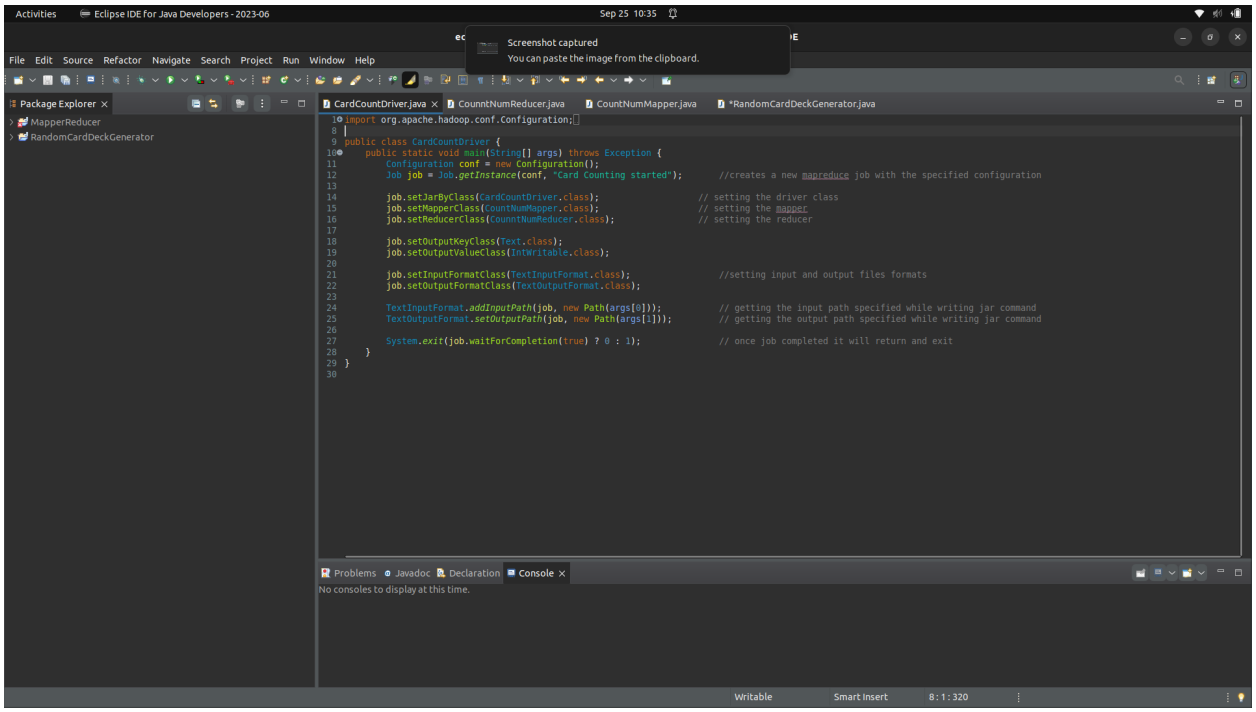
New logic:

- For all 100 decks, loop through
- Then we generate 52 cards and give it to the next for loop which will remove 17 random elements from the card deck
- This random generated elements that are removed will contain numeric as well as face cards in them
- So we have a total of 3500 cards containing both faces as well as numeric cards.



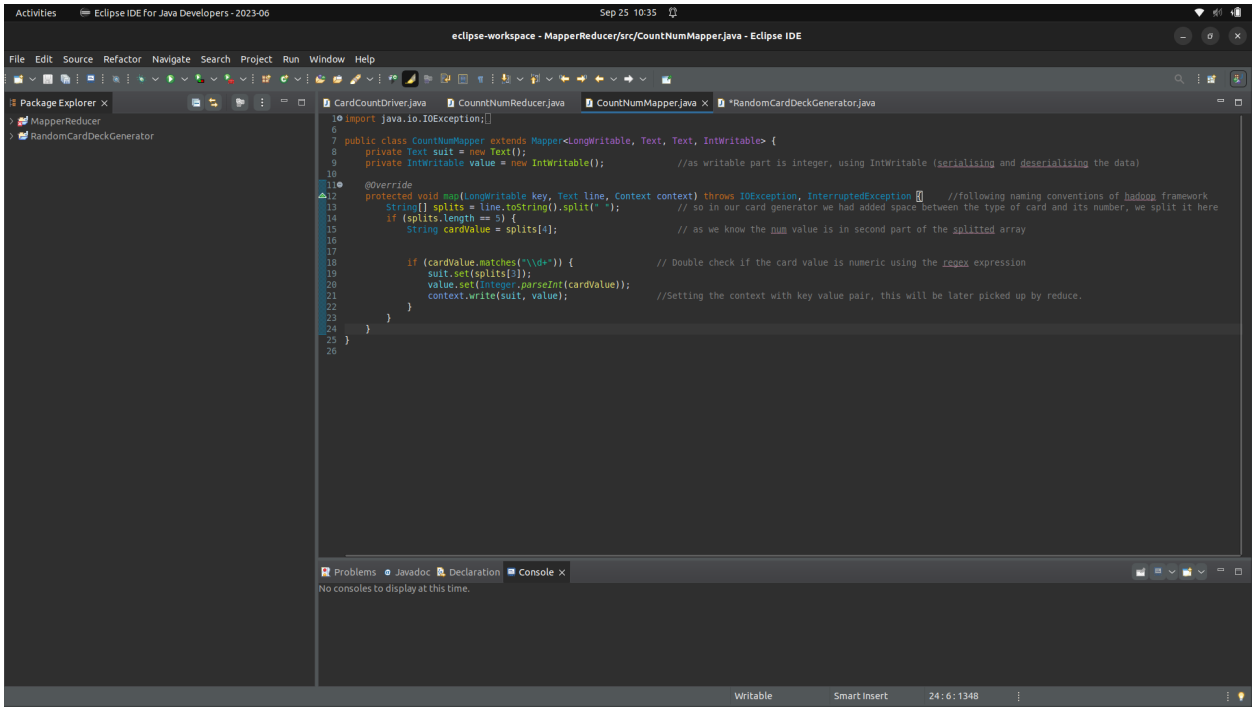
```java
import java.io.*;
import java.util.*;

public class RandomCardDeckGenerator {
    public static void main(String[] args) {
        /* This function generates card deck of given size 100/1000, then shuffles them and puts them in an input file*/
        Random random = new Random();

        List<String> total_card_list = new ArrayList<>();
        String[] weight = {"2", "3", "4", "5", "6", "7", "8", "9", "10", "Jack", "Queen", "King", "Ace"};
        String[] type_of_cardsList = {"Clubs", "Diamonds", "Heart", "Spades"};
        int maxCardDecks = 1000;                    // We can change this value as per the inputs in HW. This can be made argument based as well

        for (int deck = 0; deck < maxCardDecks; deck++) {
            List<String> cardsList = new ArrayList<>();
            for (String type : type_of_cardsList) {
                for (String value : weight) {
                    cardsList.add("Deck Number " + deck + " " + type + " " + value);
                }
            }
            for(int i = 0; i < 17; i++) {
                int ranCard = random.nextInt(cardsList.size());
                cardsList.remove(ranCard);
            }
            total_card_list.addAll(cardsList);
        }

        Collections.shuffle(total_card_list);               // Shuffle the card decks based on types and weights

        try (BufferedWriter writer = new BufferedWriter(new FileWriter("rando_input_1000.txt"))) {
            for (String card : total_card_list) {
                writer.write(card + "\n");
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Driver code to set mapper and reducer:

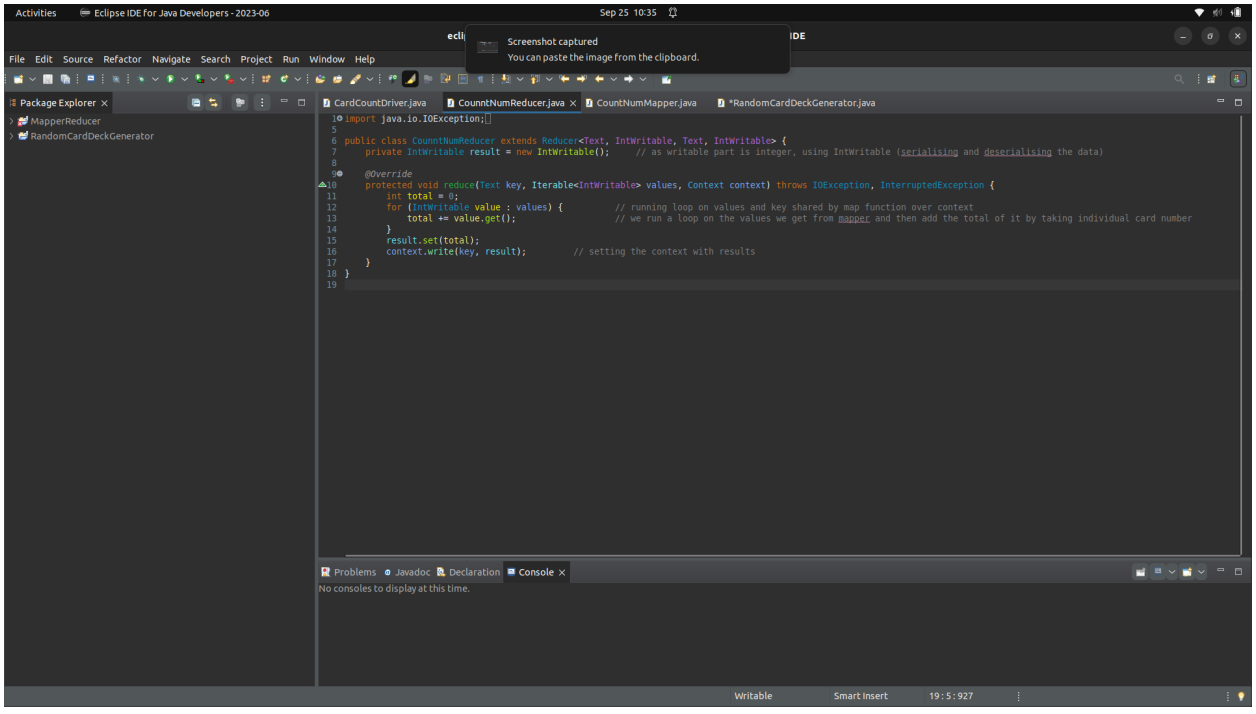


```java
import org.apache.hadoop.conf.Configuration;

public class CardCountDriver {
    public static void main(String[] args) throws Exception {
        Configuration conf = new Configuration();
        Job job = Job.getInstance(conf, "Card Counting started");       //creates a new mapreduce job with the specified configuration

        job.setJarByClass(CardCountDriver.class);                   // setting the driver class
        job.setMapperClass(CountNumMapper.class);                   // setting the mapper
        job.setReducerClass(CounntNumReducer.class);                // setting the reducer

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

        job.setInputFormatClass(TextInputFormat.class);                 //setting input and output files formats
        job.setOutputFormatClass(TextOutputFormat.class);

        TextInputFormat.addInputPath(job, new Path(args[0]));           // getting the input path specified while writing jar command
        TextOutputFormat.setOutputPath(job, new Path(args[1]));         // getting the output path specified while writing jar command

        System.exit(job.waitForCompletion(true) ? 0 : 1);               // once job completed it will return and exit
    }
}
```

Mapper Code:

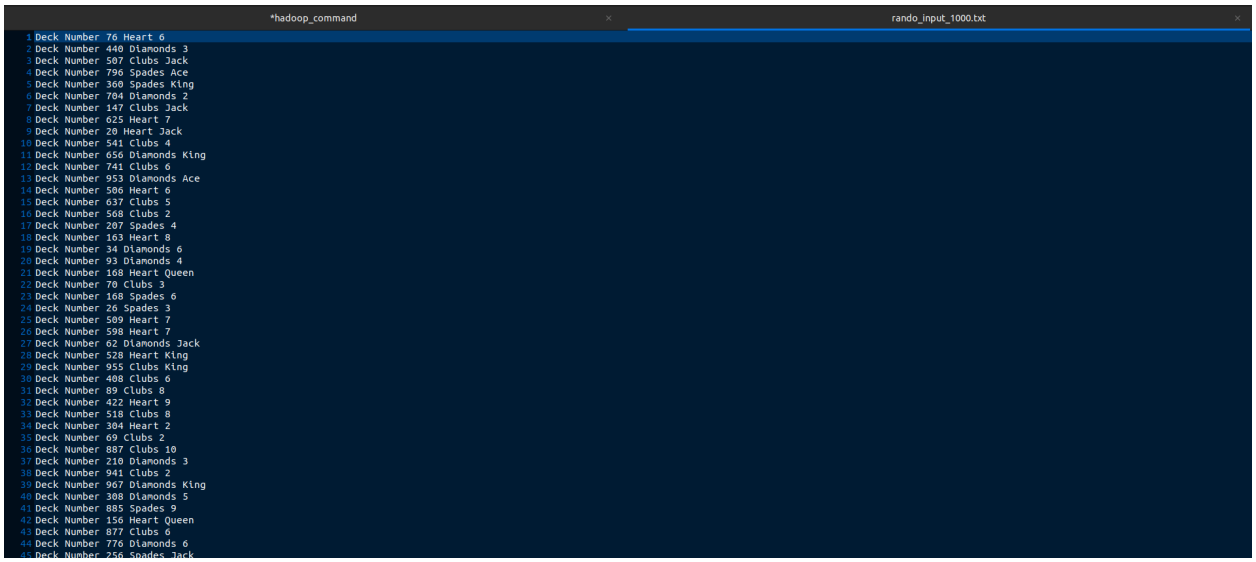


Reducer Code:



Steps:

1. Create a java file that will create our input file. This java code will fetch number of deck from the command line or predefined input
2. Move this file to hadoop → Kulkarni folder
3. Create a java project in Eclipse, the files should include driver, mapper and reducer class.
4. The driver will be responsible to set the mapper and reducer class as per the hadoop configs
5. In mapper, the code will do following things
    a. Split the input in different array
    b. Cardvalue will be stored in array[4]
    c. Check the suit value at array[3]
    d. Also check if the card is a face card or a numeric, my mapper was already doing this check so did not have to change anything in it. Just made the input change
    e. Write the splitted content in the context of hadoop which will be picked by reducer
6. In reducer, the code will make summations based on suits
    a. Run a loop to iterate over the context values set
    b. Add the sum with respect to the card suit type
    c. Set the total and generate an output file
7. A sample input file will look like below: This contains face cards as well, which will be removed in mapper later



Here the Deck number is the deck that was randomly chosen, then we have suits and then card value.

8. There are 3500 lines for 100 decks and 35000 for 1000 decks

Question 5: Try 100 decks and 1000 decks respectively?

Command used :

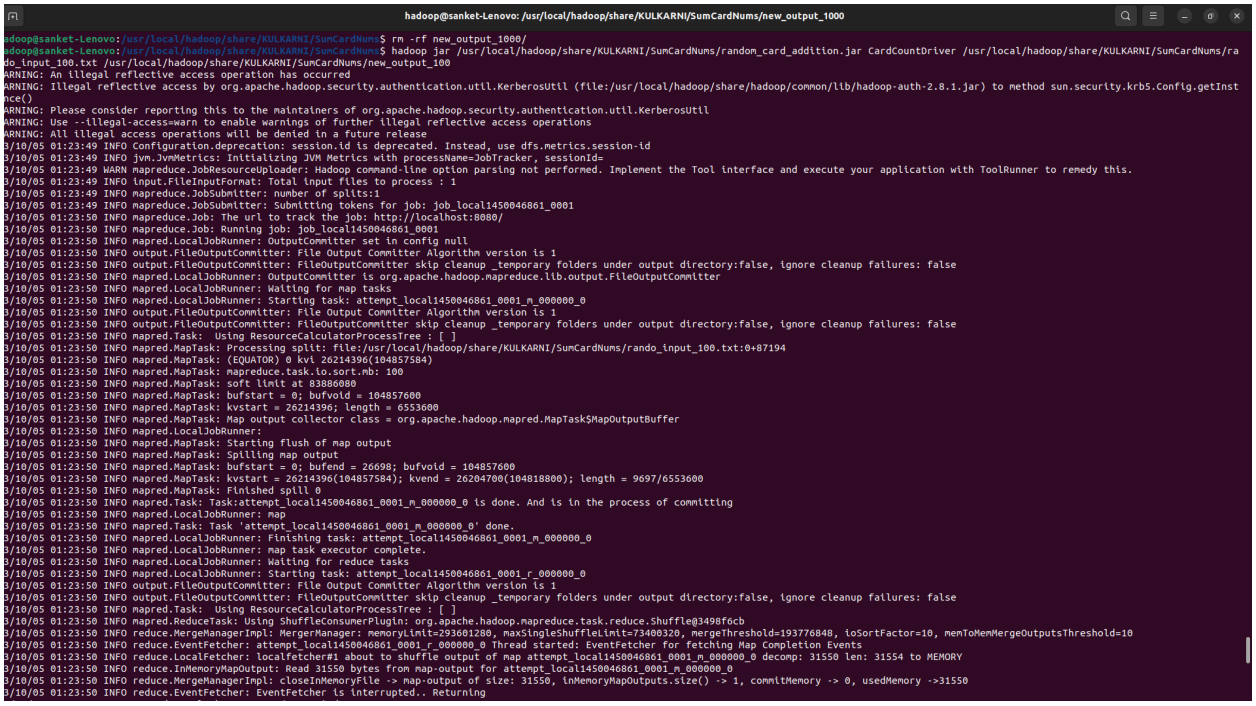
For 100 Decks

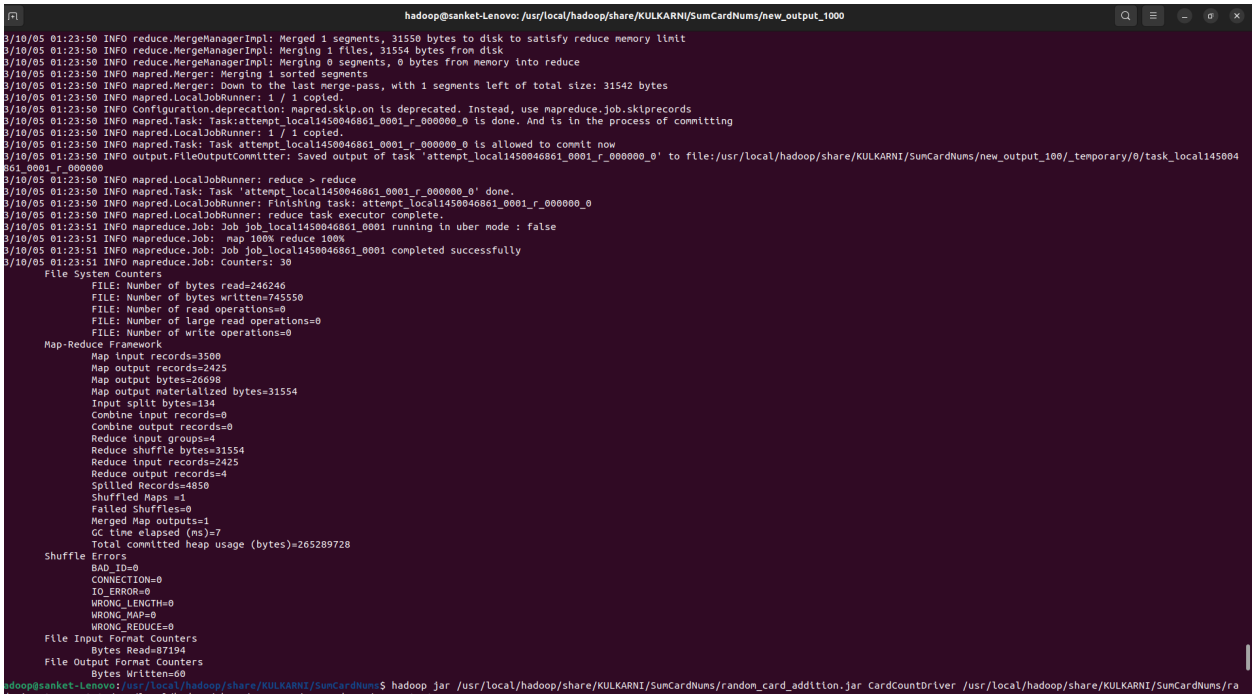
```
hadoop jar /usr/local/hadoop/share/KULKARNI/mapreduce/random_card_addition.jar CardCountDriver /usr/local/hadoop/share/KULKARNI/SumCardNums/rando_input_100.txt /usr/local/hadoop/share/KULKARNI/SumCardNums/new_output_100
```

For 1000 Decks

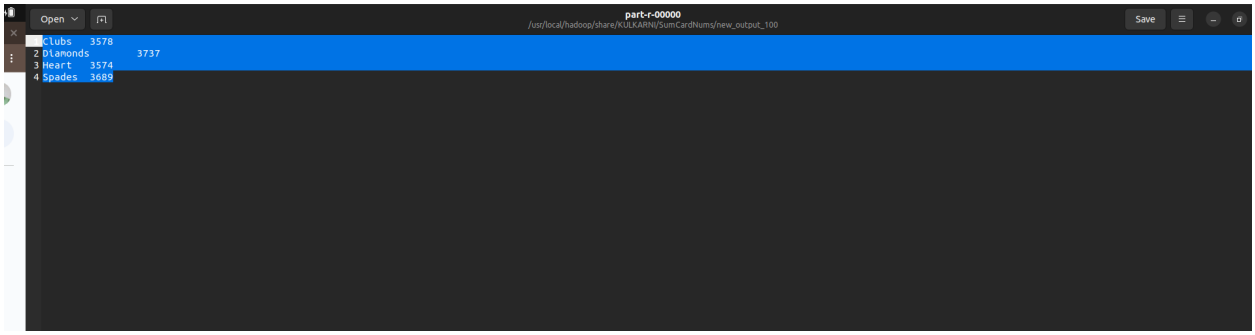
```
hadoop jar /usr/local/hadoop/share/KULKARNI/mapreduce/random_card_addition.jar CardCountDriver /usr/local/hadoop/share/KULKARNI/SumCardNums/rando_input_1000.txt /usr/local/hadoop/share/KULKARNI/SumCardNums/new_output_1000
```

Screenshots for 100 decks on new input:



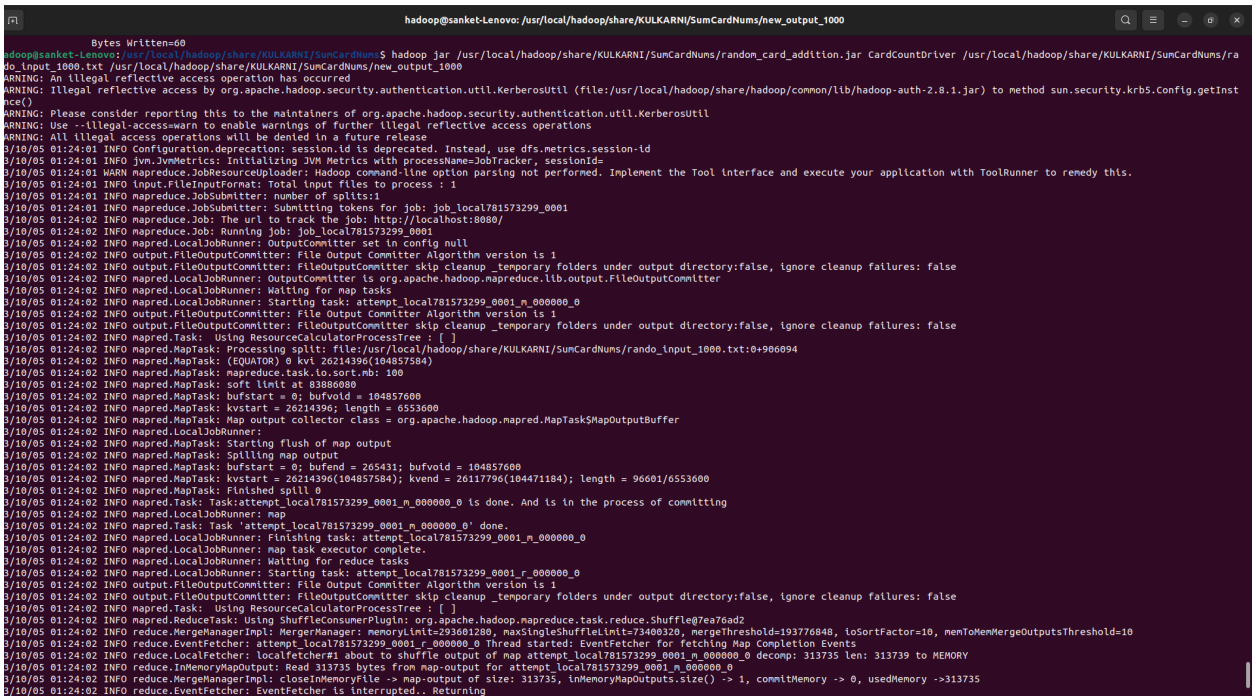


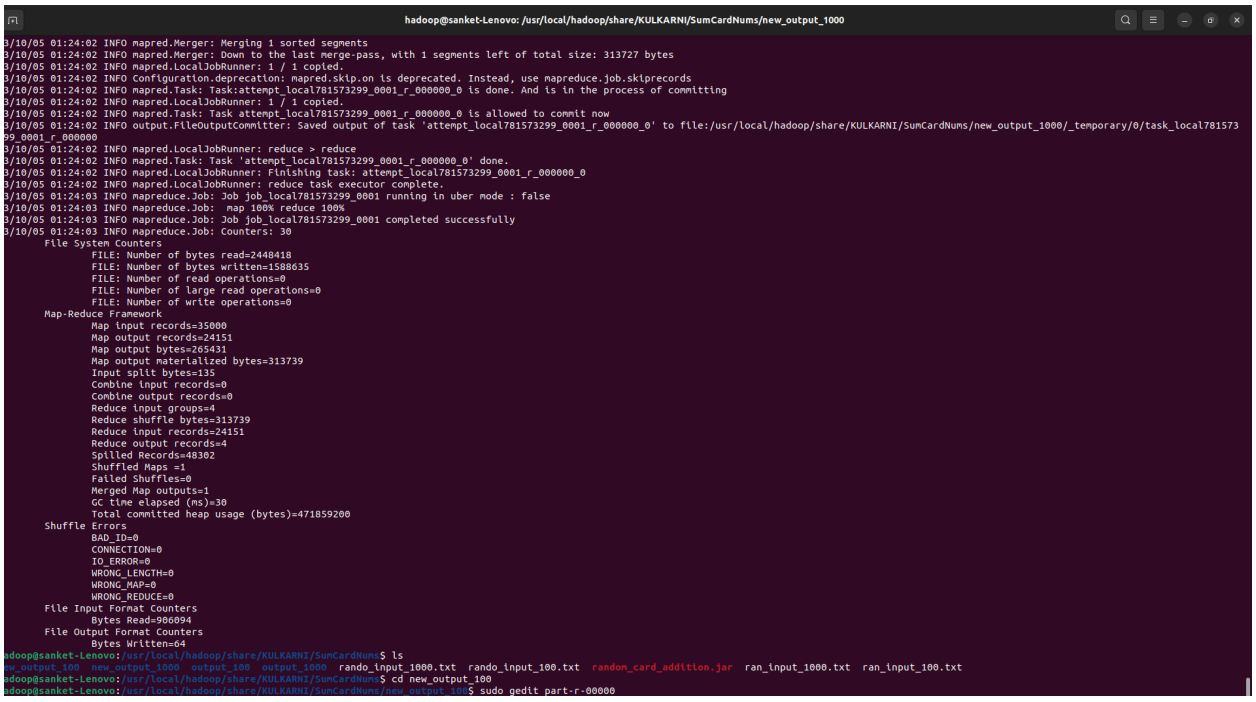
Output file for 100 deck looks like:



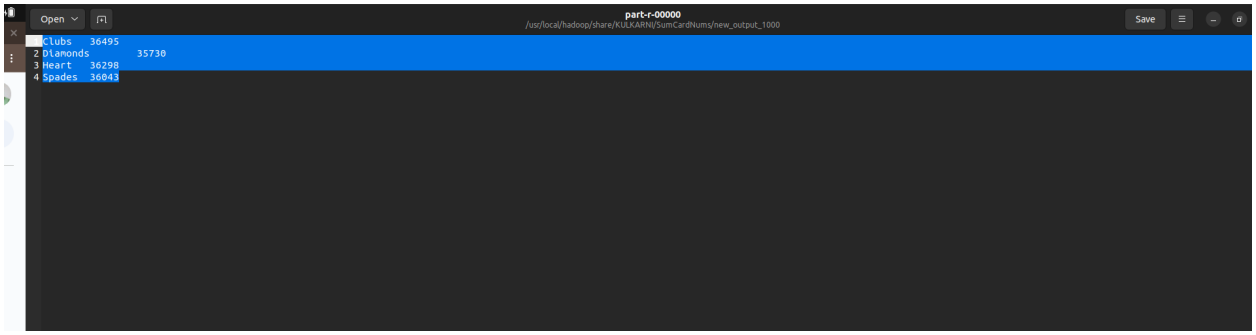
b) Trying for 1000 decks

Screenshots for 1000 decks on new input





Output file for 1000 deck looks like:



Outcomes:

Able to create a hadoop environment from scratch and setup its requirements alongside
Able to run the example jar files given in the hadoop folder by default
Able to run a python mapper and reducer using hadoop streaming and generate the output
Able to run a java mapper and reducer with driver class by creating its jar

References used:

https://www.3pillarglobal.com/insights/a-quick-set-up-guide-for-single-node-hadoop-clusters/?gclid=CjwKCAjwp_GJBhBmEiwALWBQk56GSPjAZAZPw7Qc5DqkqtQPuf_J1jIu54zNkXWsSQ7U89G6LBm4rBoC2lgQAvD_BwE
https://www.projectpro.io/hadoop-tutorial/hadoop-multinode-cluster-setup
https://hadoop.apache.org/docs/stable/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html
https://www.youtube.com/watch?v=wTkffAYsCBw
https://hadoop.apache.org/docs/stable/