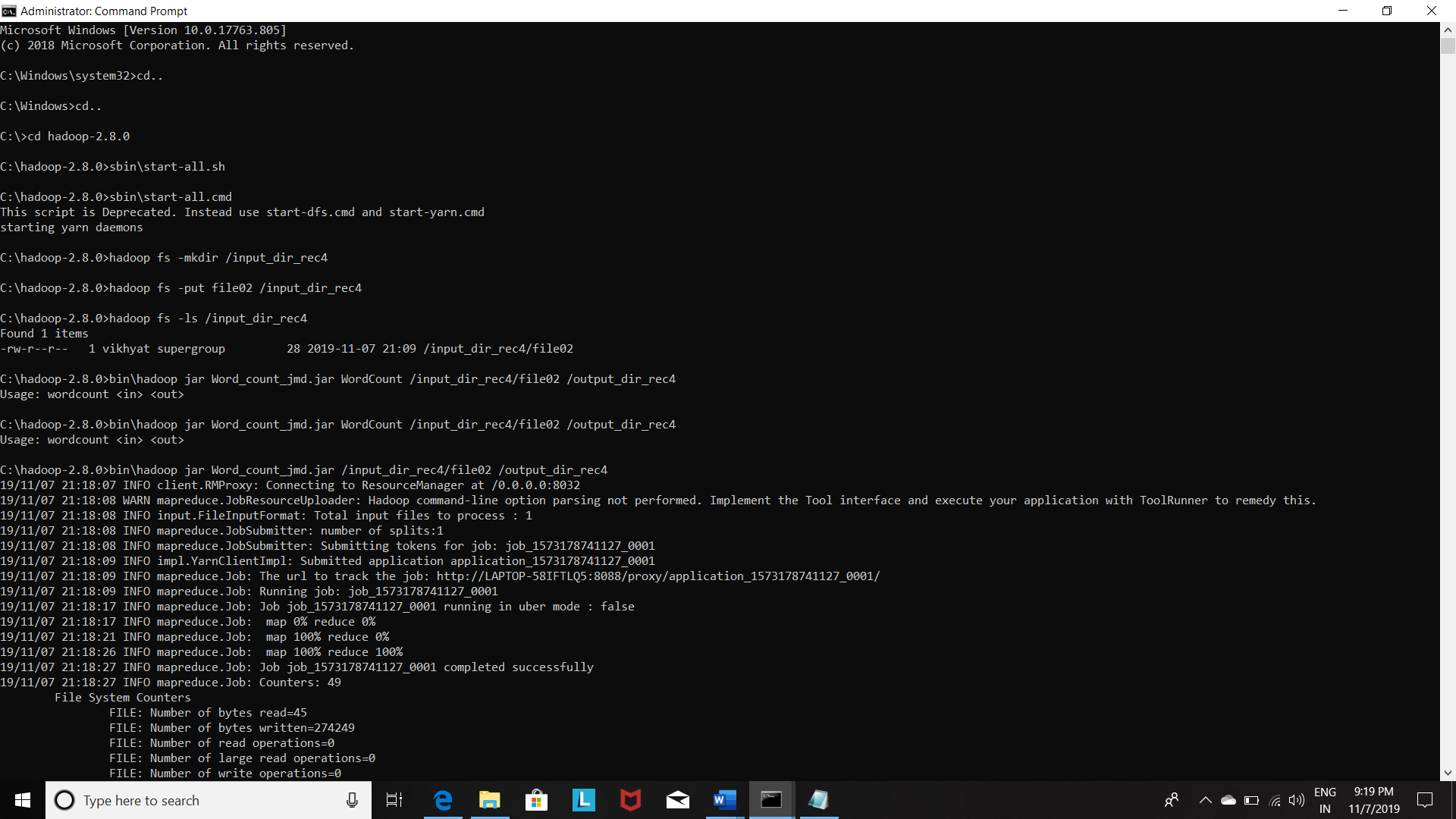
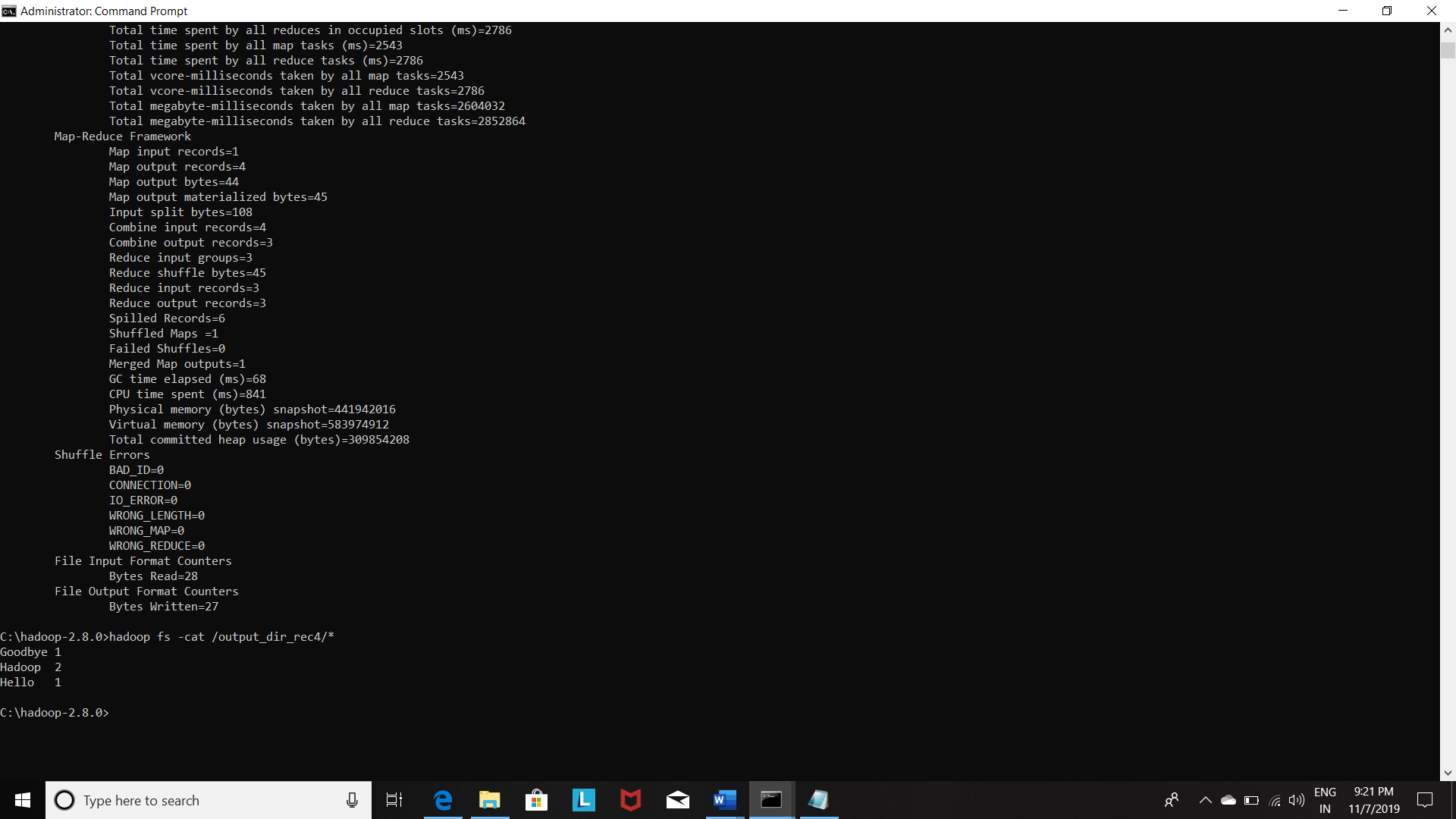
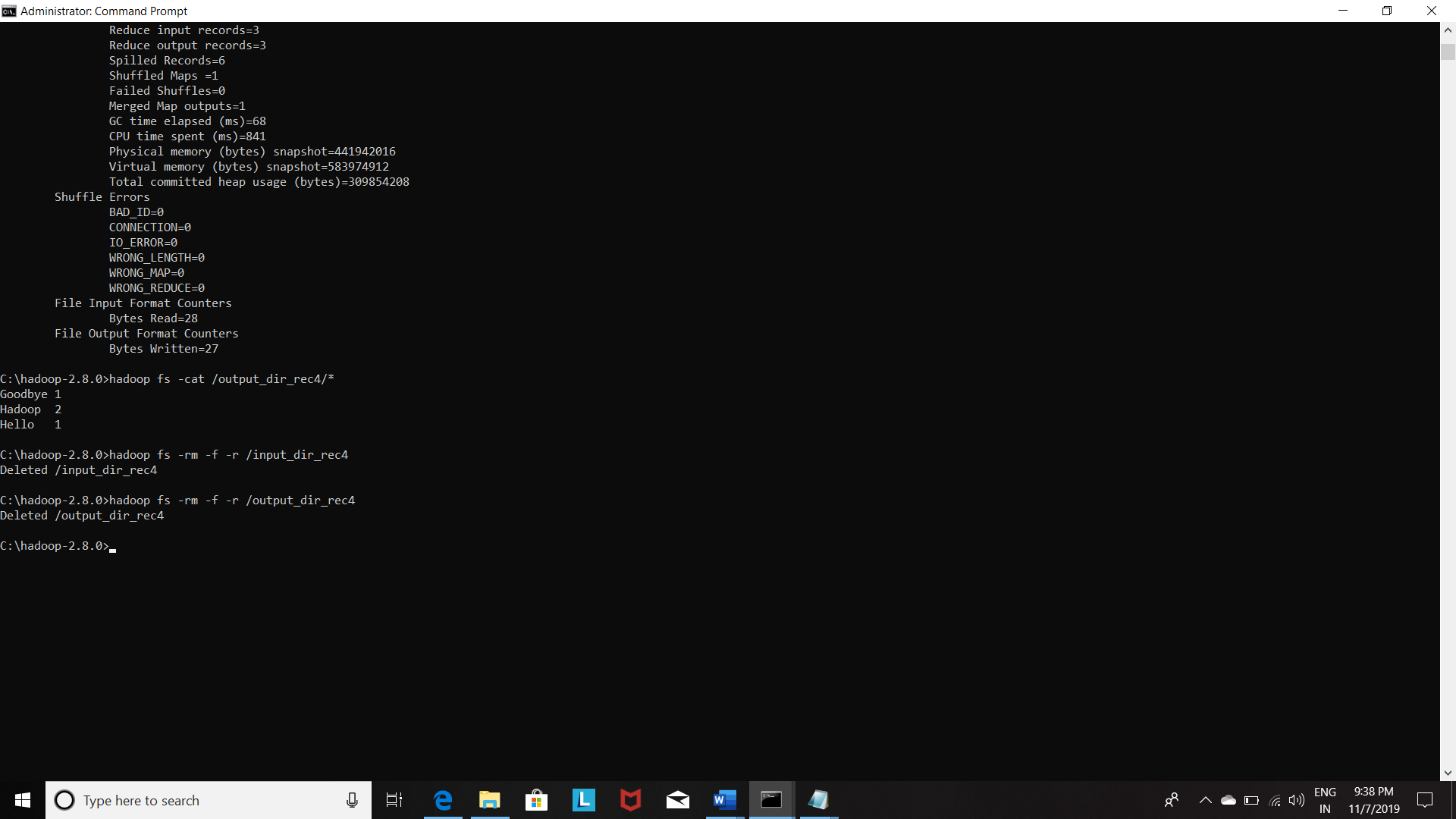
Problem 4:

1. **Word Count Java Program Execution in the Hadoop System**







**Comments:**

1. **Created the directory in Hadoop File system with the name input\_dir\_rec4 as shown.**
2. **Then we put the file file02 sample file provided inside it**

**inputdir:**

**hdfsdfs -test -e $(EXAMPLE\_DIR) || hdfsdfs -mkdir -p$(EXAMPLE\_DIR)**

**hdfsdfs -test -e $(INPUT\_DIR) || hdfsdfs -mkdir -p$(INPUT\_DIR)**

**inputs: inputdir**

**hdfsdfs -test -e $(INPUT\_DIR)/file01 \**

**|| hdfsdfs -put ../input-small/file01 $(INPUT\_DIR)/file01**

**hdfsdfs -test -e $(INPUT\_DIR)/file02 \**

**|| hdfsdfs -put ../input-small/file02 $(INPUT\_DIR)/file02**

**So above steps are being done by following commands of input directory creation and putting the input file inside it.**

1. **hadoop fs -mkdir /input\_dir\_rec4**
2. **hadoop fs -put file02 /input\_dir\_rec4**
3. **Then we used the eclipse IDE for the :-**
4. **Compilation with Hadoop libraries**
5. **Put the compiled java classes output of the compilation step into the jar file**

**Below part of the make file :**

**wordcount.jar: WordCount.java**

**mkdir -p wordcount\_classes**

**javac -classpath $(CLASSPATH) -Xlint:deprecation \**

**-d wordcount\_classes WordCount.java**

**jar -cvf wordcount.jar -C wordcount\_classes .**

1. **The resulting jar output was named Word\_count\_jmd.jar**
2. **Then we executed this job using the above jar using:**

**run: wordcount.jar inputs**

**-hdfsdfs -rm -f -r $(OUTPUT\_DIR)**

**hadoop jar wordcount.jar WordCount \**

**$(INPUT\_DIR) $(OUTPUT\_DIR)**

**Above following is the command:**

**bin\hadoop jar Word\_count\_jmd.jar /input\_dir\_rec4/file02 /output\_dir\_rec4**

**Note as suggested in the make file :WordCount main class is not mentioned in the above command (yellow color) as we put the main class name in the eclipse IDE so we need not put here in the command line for mapreduce job execution.**

1. **Job is executed and the results are being shown by the command:**

**hadoop fs -cat /output\_dir\_rec4/\***

1. **Clean Up was performed using the below commands as suggested in the make file for the deletion of directories:**

**hadoop fs -rm -f -r /input\_dir\_rec4**

**hadoop fs -rm -f -r /output\_dir\_rec4**

**Note : Hadoop used was installed in the Windows operating system hence make file execution not works here so all the steps were executed individually.**

1. **Word Count Python Program Execution in the Hadoop System**

As Hadoop is written in the Java so in order to provide the programmers the facility of programming mapper and reducer tasks in other languages the Hadoop provides the streaming jar file using which and providing the mapper and reducer executables with their path address we can execute the mapreduce job and in between these two jobs the shuffling and sorting and other tasks done in the Hadoop are being taken care by the streaming jar.

We have written the mapper.py and reducer.py which is working fine individually.

Codes:

Mapper.py

#mapper.py

import sys

# input comes from standard input STDIN

for line in sys.stdin:

    line = line.strip()

    words = line.split()

for word in words:

    print'%s\t%s' % (word,1) #Emit the word

Reducer.py

import sys

from operator import itemgetter

current\_word = None

current\_count = 0

word = None

for line in sys.stdin:

    line = line.strip()

    word,count = line.split('\t',1)

    try:

        count = int(count)

    except ValueError:

        continue

    if current\_word == word:

        current\_count += count

    else:

        if current\_word:

            print ('%s   %s' % (current\_word, current\_count))

            current\_count = count

            current\_word = word

if current\_word == word:

    print ('%s   %s' % (current\_word,current\_count))

**Further the word count using python has been done using the MRJob package i.e. used for running map reduce using Hadoop streaming.**

**File has been attached.**