Week 3 Assessment Task

Part A - MapReduce with text (6 marks)

In Part A your task is to answer a question about the data in a text file, first by writing a regular Python program, and then by writing a mapper and reducer. (By doing both you can see more readily how using a mapper and reducer differs from writing a regular program.)

When you click the panel on the right you'll get a connection to a server that has, in your home directory, the text.txt file that you have already seen, containing some sample text (feel free to open the file and explore its contents).

Your task is to **find all the words that have the largest length in the file**.

Given the example text, the output is like: (You can assume that sentences are separated by full stops and that the words within a sentence are separated by spaces.)

In this example text, the longest words are '**consecrate—we' and 'devotion—that'**, and here is the final result (all the longest words are separated by the space character):

> The longest word has 13 characters. The result includes: consecrate—we devotion—that.

In this assignment, you will use only one reducer.

First (2 marks)

Write a regular Python program to do this. A file called "regular.py" has been created for you - you just need to fill in the details.

You can test your program by running the following command (it tells Python to execute regular.py, using text.txt as input):

```
$ python regular.py < text.txt
```

Second (4 marks)

Write mapper and reducer programs to do this. Files called "mapper.py" and "reducer.py" have been created for you - you just need to fill in the details.

You can test your mapper and reducer by running the following command (it tells Python to execute mapper.py, using text.txt as input, and then pipe the result as input to reducer.py):

```
$ python mapper.py < text.txt | python reducer.py
```

To write your programs you should be able to modify programs that you have already seen in this week's content.

To test on Hadoop:

We need to copy the file text.txt into HDFS.

First, create your default working directory in HDFS:

```
$ hdfs dfs -mkdir -p /user/user
```

Next, let's create a directory in which to keep the input files of our MapReduce job. Call it "input":

```
$ hdfs dfs -mkdir /user/user/input
```

Upload the file "text.txt" into HDFS /user/user/input:

```
$ hdfs dfs -put text.txt /user/user/input
```

You can view the results:

```
$ hdfs dfs -cat output/part-*
```

To delete the output folder:

```
$ hdfs dfs -rm -r output
```

When you are happy that your programs are correct, click "Submit".

Part B - MapReduce with CSV (6 marks)

In Part B your task is to answer a question about the data in a CSV file, first by writing a regular Python program, and then by writing a mapper and reducer. (By doing both you can see more readily how using a mapper and reducer differs from writing a regular program.)

When you click the panel on the right you'll get a connection to a server that has, in your home directory, the employees.csv file that you have already seen, containing data about employees (feel free to open the file and explore its contents).

Here, again, are the fields in the file:

```
employee_id (integer)
first_name (string)
last_name (string)
email (string)
phone_number (string)
hire_date (date)
salary (integer)
```

Your task is to **find the average salary of the employees, to the nearest dollar**. You can use the function **round(**avg_salary**)**.

Given the example input file, the output is like:

> The average salary is $ 6462.

First (2 marks)

Write a regular Python program to do this. A file called "regular.py" has been created for you - you just need to fill in the details.

You can test your program by running the following command (it tells Python to execute regular.py, using employees.csv as input):

```
$ python regular.py < employees.csv
```

Second (4 marks)

Write mapper and reducer programs to do this. Files called "mapper.py" and "reducer.py" have been created for you - you just need to fill in the details.

You can test your mapper and reducer by running the following command (it tells Python to execute mapper.py, using employees.csv as input, and then pipe the result as input to reducer.py):

```
$ python mapper.py < employees.csv | python reducer.py
```

To test on Hadoop:

We need to copy the file text.txt into HDFS.

```
$ hdfs dfs -mkdir -p /user/user
```

```
$ hdfs dfs -mkdir /user/user/input
```

```
$ hdfs dfs -put employees.csv /user/user/input
```

Run the MapReduce job with 3 reducers.

```
$ hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-*.jar -file ~/mapper.py -mapper ~
```

You can view the results:

```
$ hdfs dfs -cat output/part-*
```

To delete the output folder:

```
$ hdfs dfs -rm -r output
```

To write your programs you should be able to modify programs that you have already seen in this week's content.

When you are happy that your programs are correct, click "Submit".