INF 551 – Fall 2016

Homework #5: Hadoop MapReduce & Apache Spark

[100 points]

Due: 11:59pm on 11/29/2016 to Blackboard

1. [40 points] Write a Hadoop MapReduce program "LengthCount.java" (by modifying "WordCount.java") to compute a "histogram" on the length of words in the given collection of text files.

   Assume that words fall into the following categories based on their length.

# of characters in the word	Category
10+	Big
5 to 9	Medium
2 to 4	Small
1	Tiny

Compile a jar "lc.jar" to contain the MapReduce classes (similar to wc.jar).
To simplify, in this assignment you may just use "StringTokenizer" to split input strings into "words" as seen in the WordCount.java example, and you needn't do further format checking on "real English words". For example, "inf551", "world!", "3", "abcdef" are all considered as valid "words" in this assignment. And different occurence of the same word are counted every time it appears. For example, if we have 3 "hello" (which should be a "Medium" word), we add 3 to the count of "Medium".

- Example invocation of your program:
  bin/hadoop jar lc.jar LengthCount <input> <output>

where input and output are the paths of two directories (for your own test purpose, you may use any paths).

The input directory will hold text files to be counted. Note that there might be more than 1 files in the directory, and the MapReduce framework will take in all of them and do partition automatically. In your implementation, all of the files should be counted together and the output should be counts in total rather than one output per file.

The results will be placed in the output directory by MapReduce framework. It may contain a number of files, named like "part-r-00000", one per partition (that Hadoop decides). You don't need to take care of the output format and file names yourself, just leave it to the framework. Note that when you run the code, the output directory must not exist yet, you need to just give a name to it and it will be created automatically when the code runs properly.

- Example output:

```
Big     715
Medium  701
Small   1266
Tiny    71
```

Submission: <FirstName>_<LastName>_LengthCount.java and <FirstName>_<LastName>_lc.jar

2. [60 points] Consider the following tables. Underline indicates primary key. Foreign keys are as follows. Buyer and seller of Purchase refer to name of Person table; product of Purchase refers to name of Product; and maker of Product refers to cname (company name) of Company.

   Company(cname, stockPrice, country)
   Person(name, phoneNumber, city)
   Product(name, price, category, maker)
   Purchase(buyer, seller, store, product)

   You are provided with data for these tables in the form of text files where values for a record are **comma**-separated. For each of the following questions, write a **Spark** program in Python.

   a. Find names of all people who live in “los angeles”, sorted by the name (ascending).
   b. Find names of all people who live in “los angeles” and bought products from "john", sorted by their names (ascending).
   c. Find names of all products and the country of their manufacturers, sorted by product name (ascending).
   d. Find sellers who have sold laptops but not cell phones.
   e. Find the average price of products for each maker, sorted by maker (ascending) and do not round the result.

   Sample Execution: bin/spark-submit part-a.py

   You may assume that data files for the tables are located at the same directory where your scripts are executed, and you can hard code the file names in your scripts in this task.
   For Strings, you may only consider exact matching (case-sensitive), i.e., by “john”, we mean only “john”, not “John”, “JOHN”.

   Note: You may NOT use Spark SQL in this homework. The queries themselves are very easy and you can even get the correct results by hand, but do not try to hard code the answers and print out. We will look into your code to make sure correct syntax of spark is used.

   Output Format: For each question, output one record per line in required order (if any). If more than one attributes required, separated them by comma like in (c): “productname,country”. Do not output any attributes not asked, and do not output any intermediate results.

   Submission: Submit Python scripts, one for each question, as part-a.py, part-b.py, etc. Attach your first name and last name at the beginning as usual.