Lab 3.

In this lab, we analyze a data set containing the information about the items reviewed by some users of Amazon. Specifically, we will start looking at the connections between the reviewed items, and to do this we will use a data set containing one line per reviewer. The first field of each line contains always the id of the reviewer (AXXXXXXXXX in the running example), followed by the list of all the products reviewed by her/him (BXXXXXXXXX in the sample lines reported below).

Here is a sample of the first 11 lines of such data set:

```
A09539661HB8JHRFVRDC,B002R8UANK,B002R8J7YS,B002R8SLUY
A1008ULQSWI006,B0017OAQIY
A100EBHBG1GF5,B0013T5YO4
A1017Y0SGBINVS,B0009F3SAK
A101F8M8DPFOM9,B005HY2BRO,B000H7MFVI
A102H88HCCJJAB,B0007A8XV6
A102ME7M2YW2P5,B000FKGT8W
A102QP2OSXRVH,B001EQ5SGU,B000EH0RTS
A102TGNH1D915Z,B000RHXKC6,B0002DHNXC,B0002DHNXC,B000XJK7UG,B00008DFK5
A1051WAJL0HJWH,B000W5U5H6
A1052V04GOA7RV,B002GJ9JY6,B001E5E3JY,B008ZRKZSM,B002GJ9JWS
```

For the following exercise, you can use the sample data set AmazonTransposedDataset_Sample.txt, which is available in the HDFS shared folder /data/students/bigdata-01QYD/Lab3

- /data/students/bigdata-01QYD/Lab3/ AmazonTransposedDataset_Sample.txt

A copy of AmazonTransposedDataset_Sample.txt is also available on the web page of the course.

Ex 1. “People also like…”

In this exercise, we try to build a very basic version of a recommending system. Your goal is to find the top 100 pairs of products most often reviewed (and so probably bought) together.

In this exercise, we consider two products as reviewed (i.e., bought) together if they appear in the same line of the input transposed file (the input file is /data/students/bigdata-01QYD/Lab3/AmazonTransposedDataset_Sample.txt). We ignore temporal constraints, so even if a decade or a thousand products have passed between the two reviews, we count the pair, as it represents anyway the tastes of a single user.

We suggest you to implement your application by using the template project contained in Lab3_Skeleton_with_libraries.zip. Import it in Eclipse by using the “Import Existing Projects” feature of Eclipse, by applying the same procedure you applied during the previous labs. The provided template already contains the skeleton of the Driver, Mapper, and Reducer classes. Fill out the missing parts and **decide if you need one single job or two concatenated jobs**.

Lab3_Skeleton_with_libraries.zip also contains two utility classes for your convenience. They should support you during the development of your solution.

- WordCountWritable
  - This class implement a personalized data type and can be used to store a pair (word, count), where word is a String and count is an Integer.
  - The public WordCountWritable(String word, Integer count) constructor is used to create new WordCountWritable objects.
  - String getWord() and Integer getCount() are used to retrieve the value of word and count, respectively.
  - setWord(String value) and setCount(Integer value) are used to set the value of word and count, respectively.
  - This class implements the Comparable interface. Hence, the public int compareTo(WordCountWritable other) can be used to compare objects of this class.
  - This class implements the Writable interface. Hence, it can be used as data type of the value part of the pairs emitted by a mapper or a reducer if needed.

- TopKVector<T extends Comparable<T>>
  - This class can be used to store/manage in main-memory (in a local variable) the top-k objects of a set of objects. The type of managed objects is T, where T can be an arbitrary class implementing the Comparable interface (e.g., WordCountWritable).
  - The TopKVector(int k) method is the constructor used to create TopKVector objects. Each object of type TopKVector contains an internal Vector storing only the top-k objects among the set of objects of type T that are inserted in it. Initially, this internal vector is empty (i.e., initially the top-k vector contains no objects).
  - public void updateWithNewElement(T newElement) is used to insert a new object in the internal top-k vector of the TopKVector object on which the method is invoked. This method inserts the newElement object in the internal top-k vector if and only if it is in the top-k objects. Otherwise, it is discarded. **Note that the parameter of the method must always be a new object (created using new as in the example below) otherwise the class will not work properly.**
  - public Vector<T> getLocalTopK() returns a Vector<T> containing the top-k objects associated with the TopKVector object on which this method is invoked (i.e., it returns a copy of the internal vector containing only the top-k objects among the ones inserted by using the updateWithNewElement method).

The following snippet of code shows how to use these two classes. Decide in which parts of your solution you need these two classes.

```
// An example showing how to create an object that is used to store/manage a top-3 vector
// containing objects of type WordCountWritable.
// top3 is a local variable stored in the main-memory of the application
TopKVector<WordCountWritable> top3 = new TopKVector<WordCountWritable>(3);

// An example showing how to insert 5 objects of type WordCountWritable in the top3
// local variable defined in the previous line of code.
// top3 automatically stores in its interval vector only the top-3 objects (based on the value of
```

```
// the count value) and discards the objects that are not in the top-3 set.
top3.updateWithNewElement(new WordCountWritable(new String("p1,p2"), new Integer(4)));
top3.updateWithNewElement(new WordCountWritable(new String("p1,p3"), new Integer(40)));
top3.updateWithNewElement(new WordCountWritable(new String("p2,p4"), new Integer(3)));
top3.updateWithNewElement(new WordCountWritable(new String("p5,p6"), new Integer(6)));
top3.updateWithNewElement(new WordCountWritable(new String("p15,p16"), new Integer(1)));

// How to retrieve the top-k objects from the local top3 variable
Vector<WordCountWritable> top3Objects = top3.getLocalTopK();

// Print the content of the top-3 selected objects on the standard output
for (WordCountWritable value : top3Objects) {
        System.out.println(value.getWord() + " " + value.getCount());
}

// The following is the output of this snippet of code if it is executed in a standalone
// Java application
p1,p3 40
p5,p6 6
p1,p2 4
```