**W1D3**

**Question 1. Create a Java simulator for WordCount.**

Create a WordCount class. You should be able to specify the number of Input-splits (the same as the number of Mappers) and the number of reducers.

You have already created the Mapper class. (See W1D1)

You have already created the Reducer class. (See W1D2)

WordCount class has **m** Mappers, where **m** is the number of Mappers and has **r** Reducers, where **r** is the number of reducers.

Mappers will produce their output. Then a method getPartition determines the reducer a specific key-value pair should go. For your convenience, see the getPartition method:

**public** **int** getPartition(String key){

**return** (**int**) key.hashCode() % r;

}

Keep this code inside WordCount class.

Once all values are in the Reducer, Reducer follows the steps outlined in W1D2.

**Add necessary output statements so that your output matches one shown in the attached file “TestDataOutput”.**

Please refer to the W1D3.zip file

**Question 2. Illustrate WordCount Algorithm.**

Also assume that there are three input splits:

Input split 0 : [apple lemon mango salmon wheat apple]

Input split 1 : [barley salmon apple orange carrot rice]

Input split 2 : [mango carrot lemon apple rice tuna]

Since there are three input splits, there will be three Mappers. Thus, Input split I is handled by Mapper I (I = 0, 1, 2). Assume that there are three reducers. Note that Mapper I and Reducer I run on the same machine (I = 0, 1, 2).

Further, let the partitioner assign all words less than letter ‘k’ to Reducer 0, all words greater than ‘r’ to Reducer 2 and everything else to Reducer 1.

Answer:



