1. For project a), report should describe clearly on the following topics:
   a) Project introduction.
      i. This project requires user to write a MapReduce job with multiple Reducers and create a Partitioner to determine which Mapper output piece is sent to which Reducer. We are to have a final output of 12 files, each one for 1 month of the year. These files will contain a list of IP address and the number of hits from that month. We will use 12 Reducers for 12 months. 0 being January and reducer 11 being December.

   b) Description on driver: job configuration, input format, output format, etc.
      i. Job config: This should have 12 reducer tasks set for each of the 12 months in a year, calling upon the partitioner class MonthPartitioner, our custom partitioner.
      ii. Input format: input is in the form of string.
      iii. Output format: should be in the form of ip address and the number of hits.

   c) Description on mapper: input key, value types, output key, value types, and method manipulation on the input key value pairs.
      i. Input key: LongWritable
      ii. Value type: text
      iii. Output key: ip address
      iv. Value type: text based on month
      v. Method manipulation: split the input into spaces, first field will be the ip address. Next split the associated position of date by /. Should be 1 spot for month in form of three letters. Create a list of months to refer to. Once the month is found, send back the ip address as key and the month as the value.

   d) Description on reducer: input key, value types, output key, value types, and method manipulation on the input key value pairs.
      i. Input key: text
      ii. Value type: intwritable, an integer
      iii. Output key: ip address as textobject
      iv. Value type: total number of hits of ip address on that month.
      v. Method manipulation: The occurrence of ip address in that month is totaled and value is written as keyvalue pair.

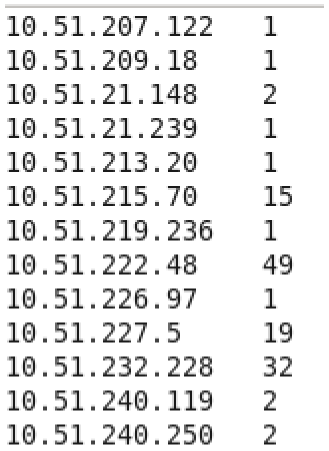
   e) Description on partitioner: how to determine which reducer should be sent for each (key, value) pair.
      i. Make each of the months as a hashmap with January representing 0 and December representing 11, 12 reducers. There must be

exactly 12 reducers. If the partitioner gets a value of “jan” or some other month, will return the number along with the ip addresses

f) Data flow description starting at the input files.
   i. Input files are compiled into a .jar file.
   ii. Job is run on cluster, input file location comes from the first argument and then given to the mapper by each line.
   iii. The mapper maps the ip address to associated month by following the custom partitioner and using 12 reducers. For the practitioner to work the job must be set to 12 reducers. The partitioner works as a condition that takes place after map phase and before reduce phase, diving data accordingly.

g) Compile procedure description.
   i. In eclipse package expore, export project and select jar file.
   ii. Choose location of the jar file.

h) Test data and results.
   i. Run map reduce job by command:$ Hadoop jar processlogs.jar stubs.ProcessLogs \weblog testlog2
   ii. View the file of testlog2 containing each month separately by running : $hadoop fs -ls testlog2
   iii. View file of month by running in format of : $hadoop fs -cat testlog2/part-r-0000….. this is month January.
   iv. Results should be in the format of ip address and the number of hits for the address. These will be separated into 12 files representing each month.



```
10.51.207.122   1
10.51.209.18    1
10.51.21.148    2
10.51.21.239    1
10.51.213.20    1
10.51.215.70    15
10.51.219.236   1
10.51.222.48    49
10.51.226.97    1
10.51.227.5     19
10.51.232.228   32
10.51.240.119   2
10.51.240.250   2
```

2. For project b), report should describe clearly on the following topics:
   a) Project introduction.
      i. Write a custom WritableComparable type that holds two strings. We count the number of times lastname/firstname pair occur within data set and output the pair and the amount of times it occurs.

b) Description on the customized writablecomparable data type, including constructors, readfields, write, compareto, equals, and hashcode methods.
   i. Contructors: empty constructor StringPairWritable() for serialization. StringPairWriteable() is constructor with two string objects provided as input(left string and right string).
   ii. Readfileds: deserializezd the fields from in by left and right.
   iii. Write: will serialize field of object write() to out by out.writeUTF(left) and out.writeUTF(right).
   iv. compareTo: compares object to another StringPairWritable object by comparing the left strings first. If they are equal, the right strings are compared. Automatically generated by Eclipse.
   v. equals: Compares two StringPairWritable object on equality.
   vi. hashcode method: generates a hascode for a StringPairWritable object. Automatically generated by Eclipse.

c) Description on driver, mapper and driver for the test job with the implemented writablecomparable type.
   i. Driver: Uses LongSumReducer, part of Hadoop API, that will sum values into a LongWritable. Will support the new StringPairWritable as a key type. The output of the class will be set to StringPairWritable.class. and the value will be of LongWritable.class.
   ii. Mapper: Mapper will split the line into words and create a new StringPairWRitable of the first two strings in that line. This pair will be the key and 1 as a value, used for summing at latter time.

d) Data flow description starting at the input files.
   i. Input files are compiled into a .jar file.
   ii. Job is run on cluster, input file location comes from the first argument and then given to the mapper by each line.
   iii. The mapper will split a line of names in to a words and emit the pair of last name and first name as a key with 1 as a value. This goes to the Reducer which will compare StringPairWritable objects by the first and last names.

e) Test data and results.
   i. Run map reduce job by command(was running into error of not finding file if I don't implement -fs=file:/// -it=local. Not sure why so had to look it up. Found on some blogpost): $ Hadoop jar writecompare.jar stubs.StringPairTestDriver -fs=file:/// -it=local ~/training_materials/developer/data/nameyeartestdata testlog3
   ii. View the file of testlog3 by running: $cat testlog3/*
   iii. Results should be in the format of (lastname, firstname) number of hits

```
(Andrews,Andrew)        1
(Andrews,Julie) 1
(Antilla,Lorenzo)       1
(Bacon,Francis) 1
(Bacon,Kevin)   1
(Bhatt,Barton)  1
(Bhatt,Jesica)  1
(Bisignano,Bronwyn)     1
(Boulton,Myriam)        1
(Braatz,Kyra)   1
(Brown,Brianna) 1
(Brown,John)    3
(Brunet,Thomas) 2
(Brunet,Yvonne) 1
(Carroll,Diana) 1
(Carroll,Lewis) 1
```