Problem 1

(a) ADRIANO 111
Whether 41
love 2221
loves 203
the 25578
whether 79
we 2922
zodiac 1

(b) 29183
$hadoop fs –cat out_wc/part-r-00000 | wc –l

(c) Improvement 1: filter out stop words such as “the” to improve both memory complexity and data quality since that type of data does not contain any useful information.
Improvement 2: make the keys (words) cases incentive to condense the output file to save memory. The word order (whether at the beginning of a line) is usually discarded in “bag of words” model and therefore cases-insensitive grouping will reduce output file size.

(d) The record reader breaks the data into key/value pairs for input to the mapper. The input for record reader is the data file to perform word-count and the output for record breaks are pairs of LongWritable/Text as LongWritable/Text is the input for map method in WordMapper.java.

(e) Each part-r file is the output of a reduce task. Each reduce tasks will executes the reducer function multiple times and write outputs to the files.

Problem 2

(a) Yes I expect there to be a significant skew in the runtime. Since we do not use a combiner at the Map tasks, some keys (words in this case) will have an extremely long list of values due to its high frequency, whereas some other keys only occur rarely and result in the reduce task finishing quickly. Therefore I would expect there to be significant skew in the runtime without using any combiners.

(b) No I do not expect there to be a significant skew in the 10-reduce-tasks scenario because in the average case I would expect the keys will be distributed out randomly and evenly, thereby each would have similar runtime.
On the contrary, I would expect there to be a significant skew in the 10000-reduce-tasks case. This is because when we are running 10000 reduce tasks, each task would only get a small number of keys and the chances are that keys of different occurrences will not be distributed out evenly. Reduce tasks assigned with infrequent terminologies will run much faster than reduce tasks assigned with common used words.

(c) No I do not expect there to be a significant skew in either scenario if combiners are used. Combiner will help to deal with the skewed data such as count-word-frequency because reduce function is associative and commutative for frequency data (the order we process the English Wikipedia articles does not matter in counting words frequency, the final frequency will always be the same). Therefore I would not expect there to be a significant skew in either scenario.

Problem 3

(a) The driver on the client first configures the job and then submits the job to the cluster. The resource manager receives the job and invokes the node manager to launch an application master process. The application master process may then request more computing power from resource manager for distributed computation by invoking more node managers for more map reduce task executions on data nodes. When the results are computed, they will be returned to client by the application master.
If possible, map tasks are executed local to the data – on the node where the block of data to be process is stored for data locality optimization. If not, map task will transfer the input data across the network. When map tasks finish, their output (intermediate data) will be stored on local disk instead of HDFS. However, reduce tasks output will be written directly to HDFS.

(b) In HDFS, files are distributed and stored on multiple nodes and the masters keeps track of the Meta data of the distributed files. Based on that schema, data locality optimization in the Hadoop means that tasks are executed on the node where the input files are stored on and the intermediate output files from map tasks are also stored on the same node. This is optimal because there is no need to transfer files around.
If the task cannot be executed locally to the data, then there will be a huge amount of file transferring from storage nodes to working nodes where the tasks are executed, which will result in really bad (long) runtimes.