Big Data Lab Answer Sheet.

Please complete this answer sheet and turn it in at the beginning of class on the due date posted in LEARN.

Part I

|  |  |
| --- | --- |
| **Part 1:** | Answer |
| **1**  (4 pts) | NOTE TO GRADER: Could you please grade my Lab 3 if it has not been done already? I submitted it weeks ago but had to resubmit the documents so that grade is missing from my grade book. I would greatly appreciate it, thank you so much!  The JSON data formatting layout is organized alphabetically by state and then by year for each different cause of death. |
| **2**  (4 pts) | The XML data formatting layout is organized by the type of death for a given state and the number of that specific type of death in a given year. All of the data is enclosed in descriptive tags like that of html, which describe the data in the tags. |
| **3**  (4 pts) | Accidental Deaths in 2015 in Alabama: 2552  Accidental Deaths in 2015 in Alaska: 388  I think that these numbers are very different given the population and the location of each state. People living in Alabama as opposed to Alaska may be exposed to more situations in which that they are more likely to die by accidental death and there are less people living in Alaska than in Alabama |
| **4**  (6 pts) | I navigated to the “Lottery Mega Millions Winning Numbers: Beginning 2002” data set by searching for XML data sets in the data nav tab. I prefer the XML format because it is easier for me personally to read. |
| **Part 2:** | Answer |
| **5**  (2 pts) | It emits the key value pair, which is the word and the number of times that it appears in the line. |
| **6**  (2 pts) | The mapper uses the individual lines of the poem as its “piece” of the data. |
| **7**  (2 pts) | MapReduce jobs are executed in this way so that large amounts of data, unlike the sample 1, can be more easily divided and split into smaller jobs for many more computers to execute. |
| **8**  (2 pts) | The MapReduce job counts and finds the key value pairs for all of the characters/letters in the data. This can be seen in the total counts for each letter in the data and also the total number of spaces as well. |
| **9**  (2 pts) | The function produces the emitted values by first splitting each line into words and storing all of the words in the var words\_list. The function then takes each word and its numeric value and stores it in the word\_counts map. If it finds that a word is already in the word\_counts map, the value is incremented to add one, otherwise if it is a new word the value is stored in the word\_counts map as being 1. The key value pair is then printed as being the word in the word\_counts map and the total count of each word in the word\_counts map. |
| **10**  (2 pts) | The reducer uses each word in the data as its input and then it emits  the word in the data : its value in the data |
| **11**  (5 pts) | This still works because although the mapper only lists each word and gives it a value of 1 each time it encounters a word, the reducer is able to find the totals of each word in the data that the mapper found and gave a value of 1. |
| **12**  (15 pts) | Alice 81.6666666667  Bob 68.0  Carol 67.0  Dave 78.0  Eve 63.6666666667 |
| **13**  (20 pts) | def mapper(key, value):  info\_map = eval(key)  for field in info\_map:  if field == 'Enrollments' or field == 'Dropouts':  field\_value = info\_map[field]  Wmr.emit(field, field\_value)    def reducer(key, values):  sum = 0  for value in values:  sum = sum + float(value)  Wmr.emit(key, sum) |
|  | NOTE TO GRADER: Could you please grade my Lab 3 if it has not been done already? I submitted it weeks ago but had to resubmit the documents, so that grade is missing from my grade book. I would greatly appreciate it, thank you so much! |