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) | The data is organized by placing variables on the left with the data to the right of it, it goes down vertically for quite a while showing what I believe may be the individual data |
| **2**  (4 pts) | The data is organized by states where they show the amount of deaths per year as well as the leading cause of deaths. The states are organized alphabetically |
| **3**  (4 pts) | In Alabama there were a recorded 2755 deaths and in Alaska there were 439 recorded. I think these numbers are so different because there are over 4 times the amount of people living in Alabama than Alaska |
| **4**  (6 pts) | The data I choose to look at was crime data dating from 2010 to the present. The format I prefer is the XML format, I find it a lot more straight forward than the Json format and easier to understand. I navigated it by sorting through the XMS format since I knew I liked that one much more and searched for local government since I thought it could be interesting. |
| **Part 2:** | Answer |
| **5**  (2 pts) | It appears it counts each word in a line and how many times it pops up in said line. So, in line 3 of the example it counts “the” twice, but “tune” is only counted once. |
| **6**  (2 pts) | It uses the entire poem as the data counting line by line until it reaches the end. The pieces is each step broken up by lines. |
| **7**  (2 pts) | It breaks the data up so it can assign each piece to a computer and has each one complete their equal tasks in order to be the most efficient |
| **8**  (2 pts) | Rather than splitting up each data by spaces it makes the whole line into one piece of data and counts the letters in the line instead |
| **9**  (2 pts) | First the data takes the list and splits up each word at every space then it records every word and in the list setting it equal to one, adding +1 each time the word appears in the line. The bottom part makes sure that the word does not pop up twice in the list. |
| **10**  (2 pts) | The reducer function counts the total amount of times the words pop up in the entire poem, since it uses the entire poem it will not give data until the entire function has been run. It will emit values like “the = 9” |
| **11**  (5 pts) | This approach still works because you’re still counting every time the word pops up it still records it in the list. It’s less user friendly in my opinion but still works all the same. |
| **12**  (15 pts) |  |
| **13**  (20 pts) | def mapper(key, value):  grade\_map = eval(key) # automatically parses the data in JSON format  # grade\_map['Alice'] contains 95 87 or 63 (and so on) depending on which row the mapper is reading    for student in grade\_map: # student is 'Alice', 'Bob', and so on  # get each of the 4 grades in your mapper data line  if student == "Dropouts":  dropouts = grade\_map[student]  Wmr.emit(student, dropouts)  if student == "Enrollments":  enrollments = grade\_map[student]  Wmr.emit(student, enrollments)    def reducer(key, values):  sum = 0  count = 0  for value in values:  sum = sum + float(value)  count = count + 1  if count > 0:  average = sum  Wmr.emit(key, average) |