**HW1 Rubrics**

**DSCI 551, Fall 2022**

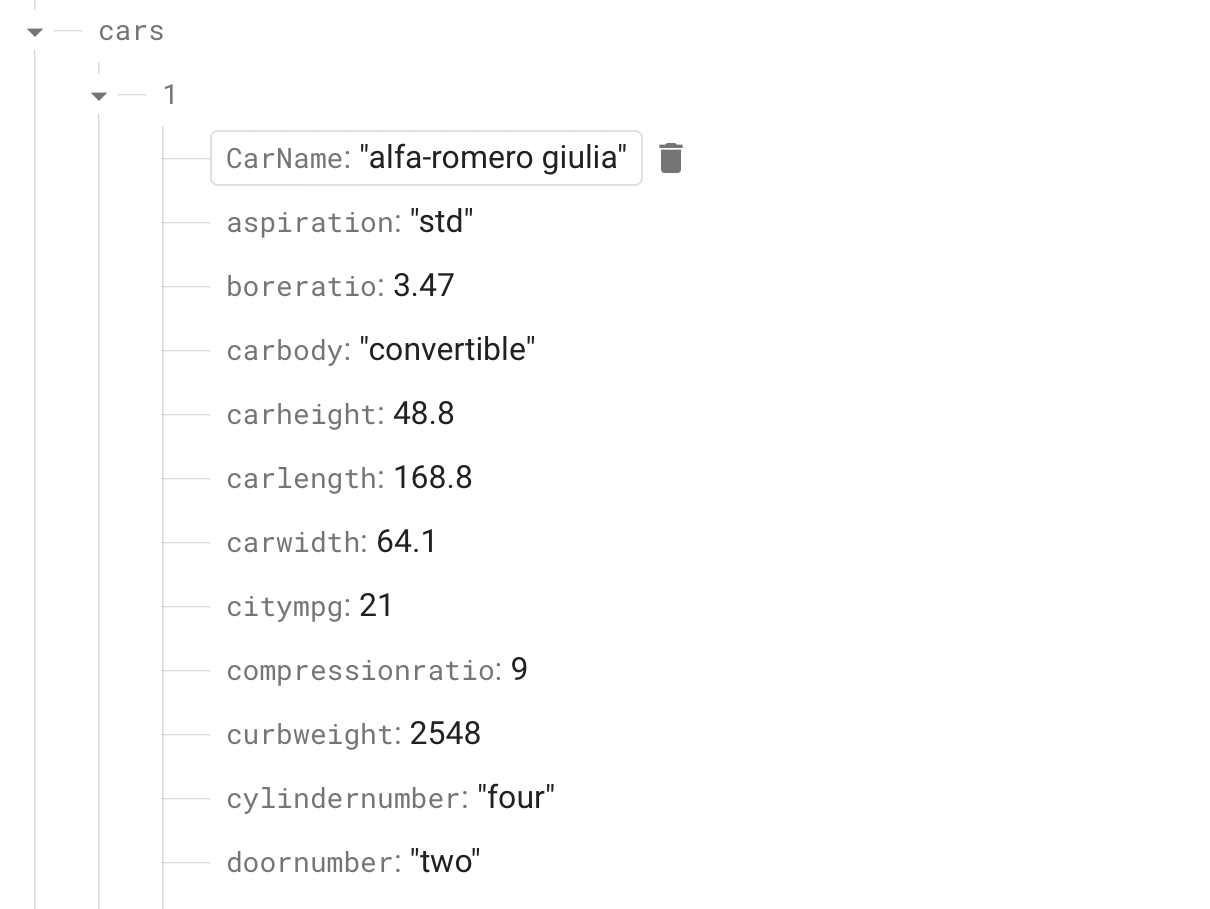
**100 points**

1. [20 points] Write a Python script load.py that takes the CSV file cars.csv and loads it to your Firebase real time database. **Note: you need to load the entire data set to Firebase. Every car should be stored as a JSON object in Firebase.**

              Execution format: python3 load.py cars.csv

GRADING:

* 10 pts: If code run else 0pts
* 10 pts: For the correctness of the data format in the screenshot. (0 for missing screenshot)
* -1 if the input is not taken from arguments and hardcoded cars.csv
* -5 if they did not change the rule but using firebase admin



1. [20 points] Search by price range: write a Python script search\_price.py that takes a range of price (in two arguments) and outputs IDs of cars in the range (inclusive). **Note: you need to create an index on car’s price in the Rules of your Firebase database. If no cars are in the specified price range, then please return ‘No cars found with the given range’.**

              Execution format (example): python3 search\_price.py 15000 16000

GRADING:

* 10 pts: If code run else 0pts
* 2.5pts: For each test case (-1 if the output message No cars found is not displayed)
* If output is not in the list format (-2) (if printed one by one, 0.5 for each case)

Test Cases:

1. python search\_price.py 15000 16000

IDs for the car price range are:

[6, 59, 112, 135, 136, 180, 181, 182, 197]

1. python search\_price.py 1500 1600

No Car\_ID with the given range

1. python search\_price.py 16000 37000

IDs for the car price range are:

[1, 2, 3, 5, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 18, 48, 49, 50, 66, 67, 68, 69, 70, 71, 72, 73, 76, 105, 106, 107, 113, 114, 115, 116, 117, 118, 126, 127, 128, 130, 137, 138, 173, 179, 198, 199, 200, 201, 202, 203, 204]

1. python search\_price.py 1500 160000

IDs for the car price range are:

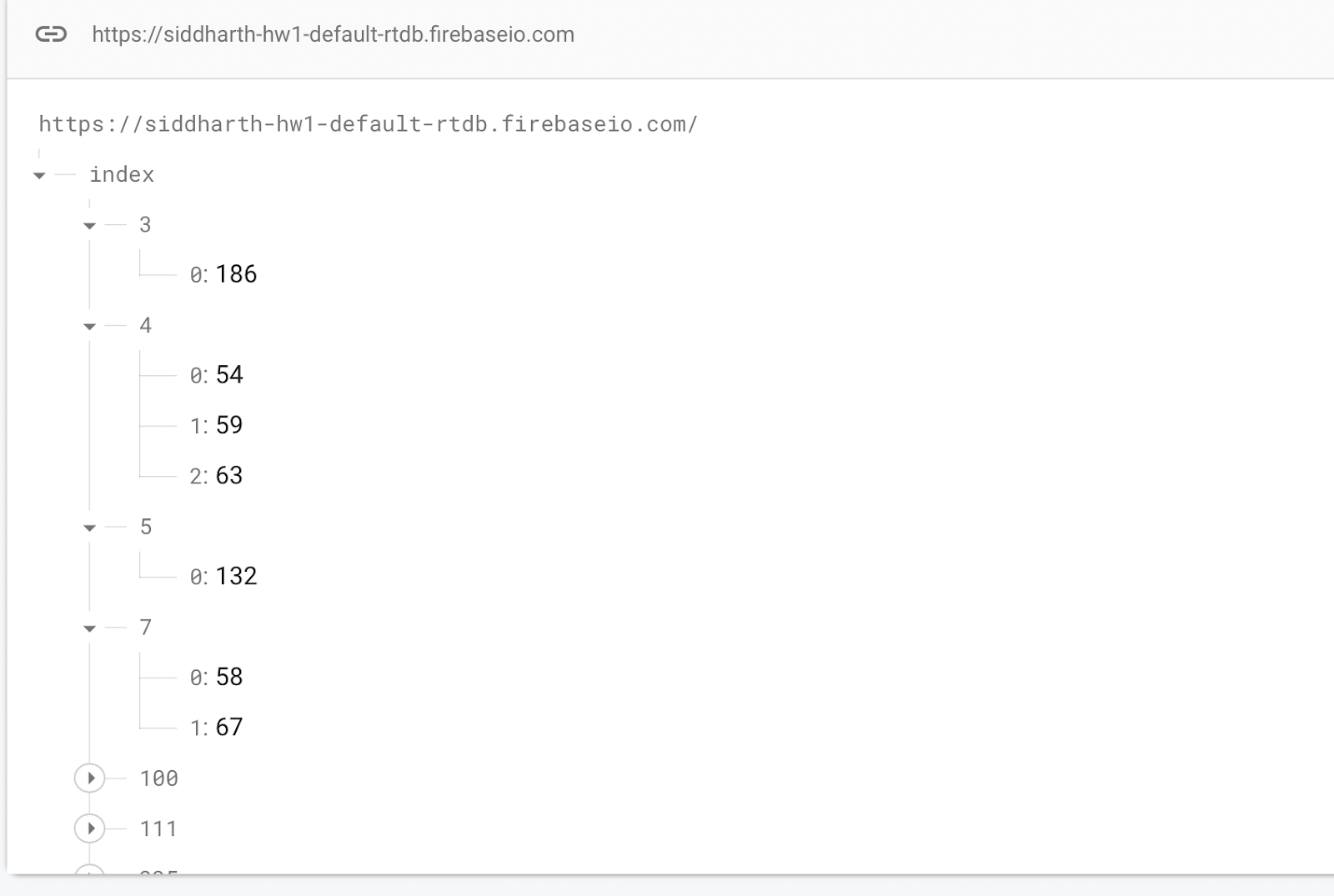
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155, 156, 157, 158, 159, 160, 161, 162, 163, 164, 165, 166, 167, 168, 169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181, 182, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194, 195, 196, 197, 198, 199, 200, 201, 202, 203, 204, 205]

1. [30 points] Create a keyword index: write a Python script create\_index.py that creates a keyword index for keywords appearing in car name. The keywords in a car name are a list of words resulting from tokenizing the car name by punctuation characters and white spaces. For example, “alfa-romero giulia” has 3 tokens: alfa, romero, and giulia, while “dodge colt (sw)” has tokens: dodge, colt, and sw. For each unique keyword in the car names, your index should store a list of IDs of cars whose name contains the keyword. You need to store the index in Firebase which will be used in the following search task. You should store index separately from the cars data set and should not overwrite the cars data set in Firebase. Note keywords are not case sensitive. So honda and Honda are considered to be identical.

              Execute format: python3 create\_index.py cars.csv

GRADING:

* 15 pts: If code run else 0pts
* 15 pts: For the correctness of the data format in the screenshot. (0 for missing screenshot)
* -1 if the input is not taken from arguments and hardcoded cars.csv or if data is taken from firebase



1. [30 points] Search cars by keyword: write a Python script search\_car.py that finds IDs of cars using a list of keywords on car names. Note that search is case insensitive. Show the IDs of cars such that the cars whose name contains more keywords are shown first. You can break the ties arbitrarily. **Note the input may contain punctuation characters (e.g., entered by mistakes by users).**

               Execution format (example): python3 search\_car.py “honda accord”

GRADING:

* 15 pts: If code run else 0pts
* 3 pts: For each test cases (-1 pts for incorrect order of IDs for each test case) (-1 if just output message No cars found is not displayed and doesn't throw error, if error then -3)
* If output is not in the list format (-2) (if printed one by one, -0.5 for each case)

Test Cases:

1. python search\_car.py "honda accord"

IDs of the car are:

[41, 38, 36, 34, 43, 42, 40, 39, 37, 35, 33, 32, 31]

2.python search\_car.py "dodge coronet custom (sw)"

IDs of the car are:

[30, 28, 122, 29, 27, 25, 201, 195, 193, 187, 155, 110, 69, 68, 62, 61, 26, 24, 23, 22]

3.python search\_car.py "dodge colt hardtop"

IDs of the car are:

[26, 27, 154, 30, 29, 28, 25, 24, 23, 22]

4.python search\_car.py "dodge colt'/-'hardtop"

IDs of the car are:

[26, 27, 154, 30, 29, 28, 25, 24, 23,22]

5. python search\_car.py "kljfddfjkl"

No cars found.