A

Project Report

on

"Electric Car Traveler"

Prepared by

885185645 | Devashri Manepatil | devashri@csu.fullerton.edu

885209536 | Apoorva Machale | apoorvamachale@csu.fullerton.edu

886679687 | Sumit Bishnoi | sumitbishnoi@csu.fullerton.edu

A Report Submitted to

Prof. Doina Bein

Department of Computer Science

College of Engineering and Computer Science (ECS)

California State University, Fullerton (CSUF)

for the Fulfillment of the Requirements for the

CPSC 535: Advanced Algorithms – Project1

California State University, Fullerton
CA - 92831
October 2022

CHAPTER 1
SUMMARY

Electric Car Traveler algorithm states the problem that while traveling through the cities, we need to stop to charge our electric car if its charging is not enough to travel the next city. Also, we will have to refill at the previous station if the current station is not working. We have solved the algorithm in python language. The algorithm gives output in a list ‘result’ for the cities we had to stop to charge the electric car. This will help us when we plan long travel routes and help us identify the cities to stop to charge our electric car.

CHAPTER 2
PROBLEM STATEMENT

Given a capacity *C* in miles that represents the maximum number of miles your electric car can drive, *n* cities and (*n*-1) distances between two consecutive cities, design an algorithm that outputs the list L of cities where one need to stop and charge the car such that:

1. L is if minimum length among all possible list of cities
2. the starting city, which is the first city, is the first element of L
3. The destination city, which is the last city, is the last element of L
4. If j and k are two consecutive cities in L, then when the car is in city j, the car is able to drive to city k and back to **the city before** city **k**, in case the charge station in city k is broken.

The capacity *C* in miles is not fixed, but one can assume that is a positive integer between 250 and 350. The number of cities *n* is not fixed, but you can assume that it is greater than 3 and less than 20. The distance between cities is a positive integer and always less than C/2 and greater than 10.

CHAPTER 3
PSEUDOCODE

Step 1: Take input from the user for the total number of miles(c) and the number of cities(n) following the given constraint else print message that it is invalid input and again take input.

Step 2: Take distance input for cities starting from A to n (last city) following the given constraint else again take the distance and in the list result store the first city

Step 3: Store the distances for the respective cities like a dictionary. For e.g. {‘A’: 5, ‘B’: 10,’C’: 0} i.e.

   a. Distance from A to B is 5
   b. Distance from B to C is 10
   c. Distance for city C is 0 as it is the last city.

Step 4: Consider current_mile = c (total miles)

Step 5: While applying the loop for the dictionary we will check if we have sufficient distance to traverse to the next city by checking if the current_mile > 2*mile of the current city.

   a. If true, then current_mile = current_mile – mile for the current city
   b. Else, append the city in the result list as we will have to refill at that station.
      i. If we have current_mile less than 100, then we refill entirely i.e. current_mile = c.
      ii. Else if we have (current_mile + mile for the current city > 2* mile for the current city) will refill (current_mile + mile for the current city)
      iii. Else if we have (current_mile + mile for the current city < 2* mile for the current city) will refill (current_mile + 2*mile for the current city)

CHAPTER 4
PROJECT SETUP GUIDELINES

Method-1: If you have any IDE (VSCode/ Pycharm)

1. Download the file algo.py

2. Install the python compiler on the IDE

3. Open the file in IDE and click on run

4. You can enter the input details and get the desired output in Terminal of the IDE

Method-2: If you are using Terminal (Command Prompt)

1. Download the file algo.py

2. Open command prompt

3. Go to the location of downloaded file

5. Run the file using command: *python algo.py*

Method-3: If you are using Online Compiler (GeeksForGeeks/Coding Minutes)

1. Download the file algo.py

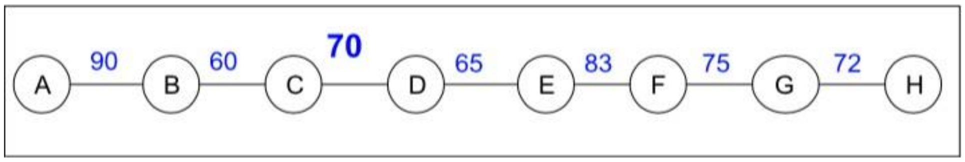
2. Copy and paste the code on the online compiler

3. Execute the code, you can enter the input details and get the desired output in the Terminal

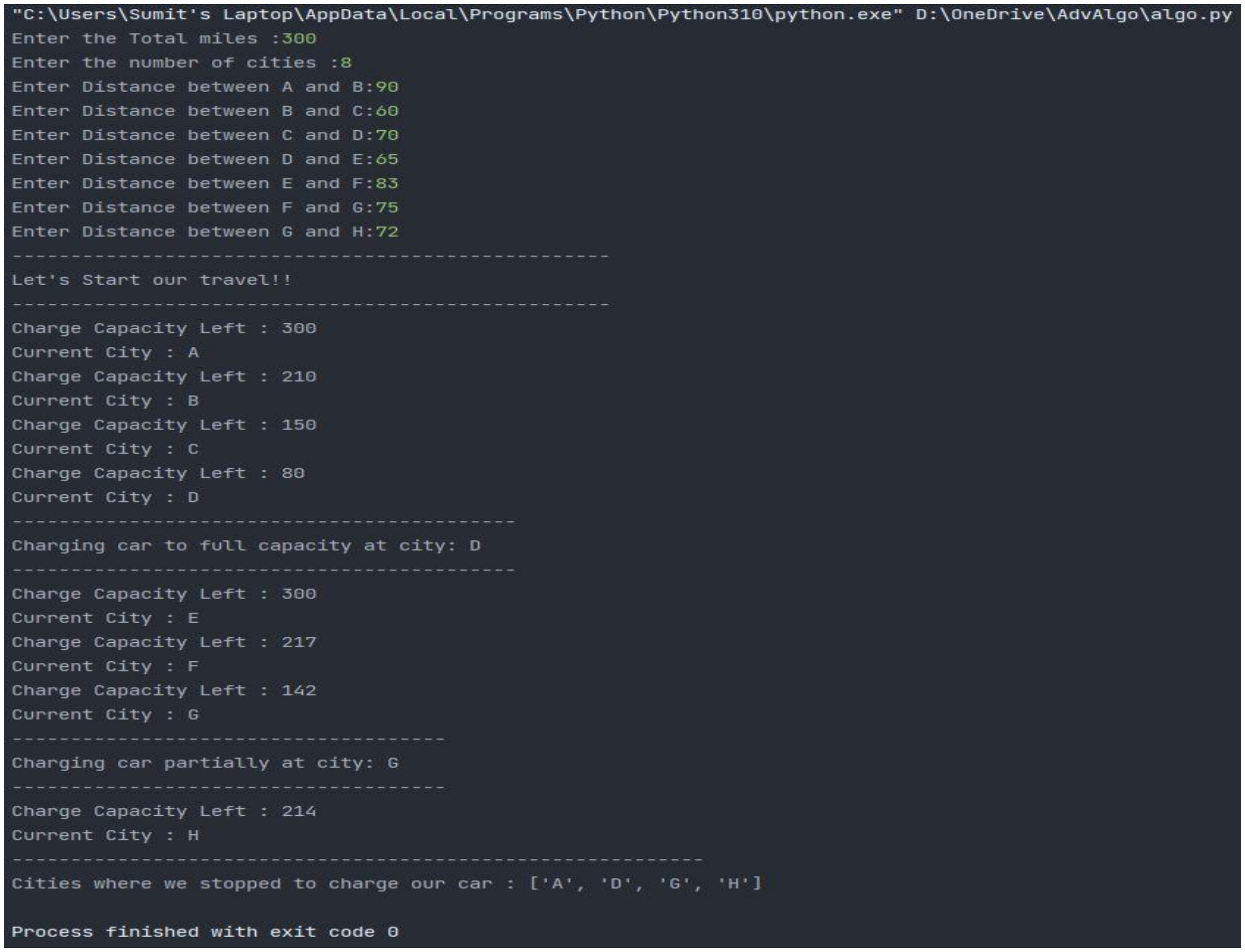
CHAPTER 5
SCREENSHOTS

Example 1

Input: The graph below in Figure 1, the capacity is C = 300 miles, the starting city = A and the destination city = H, the miles between cities are shown with blue ink.



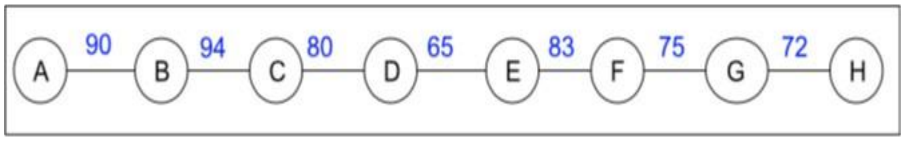
Output:



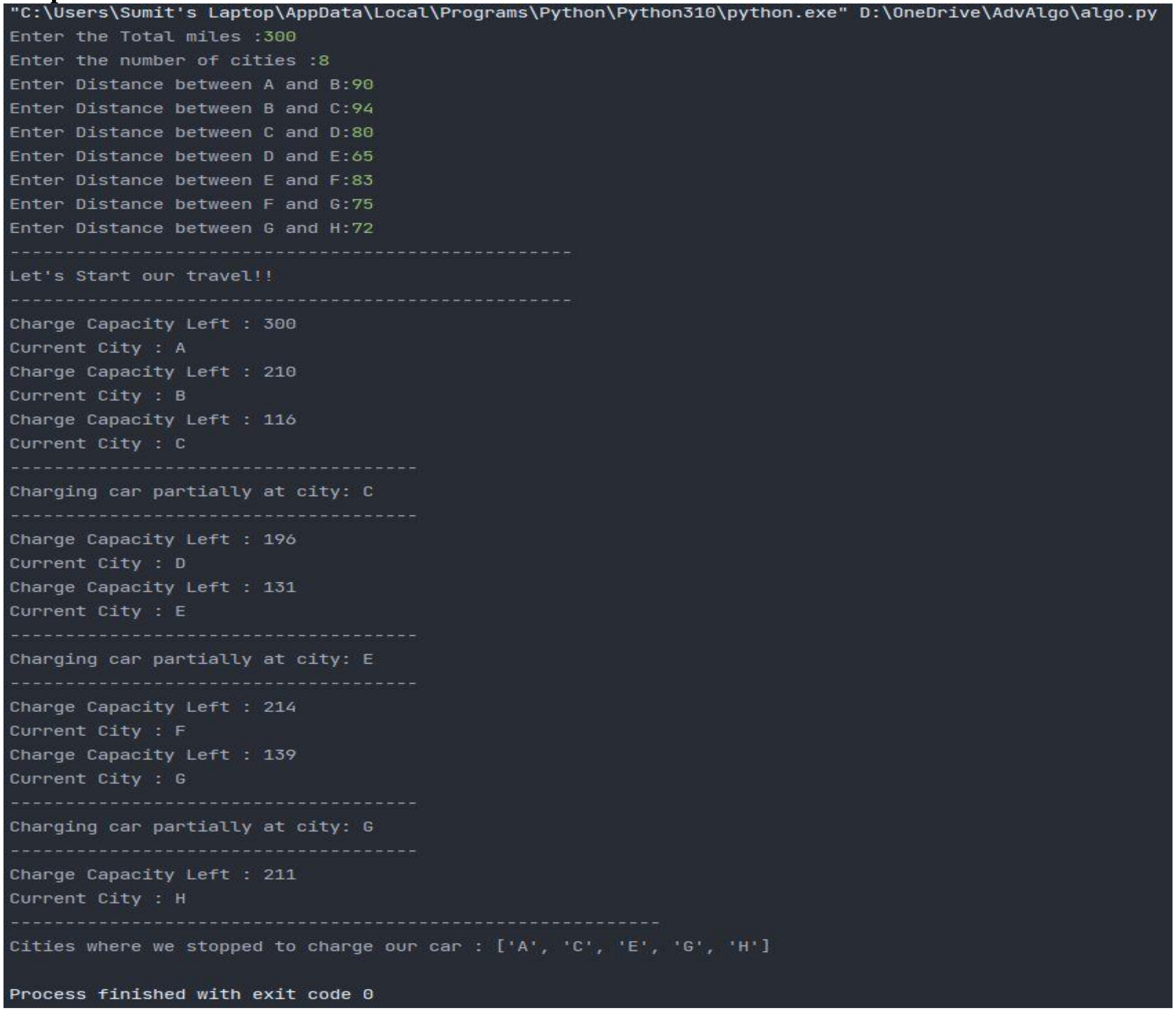
```
"C:\Users\Sumit's Laptop\AppData\Local\Programs\Python\Python310\python.exe" D:\OneDrive\AdvAlgo\algo.py
Enter the Total miles :300
Enter the number of cities :8
Enter Distance between A and B:90
Enter Distance between B and C:60
Enter Distance between C and D:70
Enter Distance between D and E:65
Enter Distance between E and F:83
Enter Distance between F and G:75
Enter Distance between G and H:72
---------------------------------------------------
Let's Start our travel!!
---------------------------------------------------
Charge Capacity Left : 300
Current City : A
Charge Capacity Left : 210
Current City : B
Charge Capacity Left : 150
Current City : C
Charge Capacity Left : 80
Current City : D
-------------------------------------------
Charging car to full capacity at city: D
-------------------------------------------
Charge Capacity Left : 300
Current City : E
Charge Capacity Left : 217
Current City : F
Charge Capacity Left : 142
Current City : G
-------------------------------------
Charging car partially at city: G
-------------------------------------
Charge Capacity Left : 214
Current City : H
-----------------------------------------------------------
Cities where we stopped to charge our car : ['A', 'D', 'G', 'H']

Process finished with exit code 0
```

Example 2

Input: The graph below in Figure 1, the capacity is C = 300 miles, the starting city = A and the destination city = H, the miles between cities are shown with blue ink.



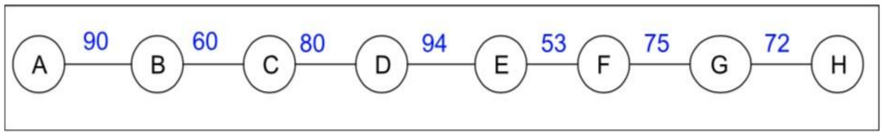
Output:



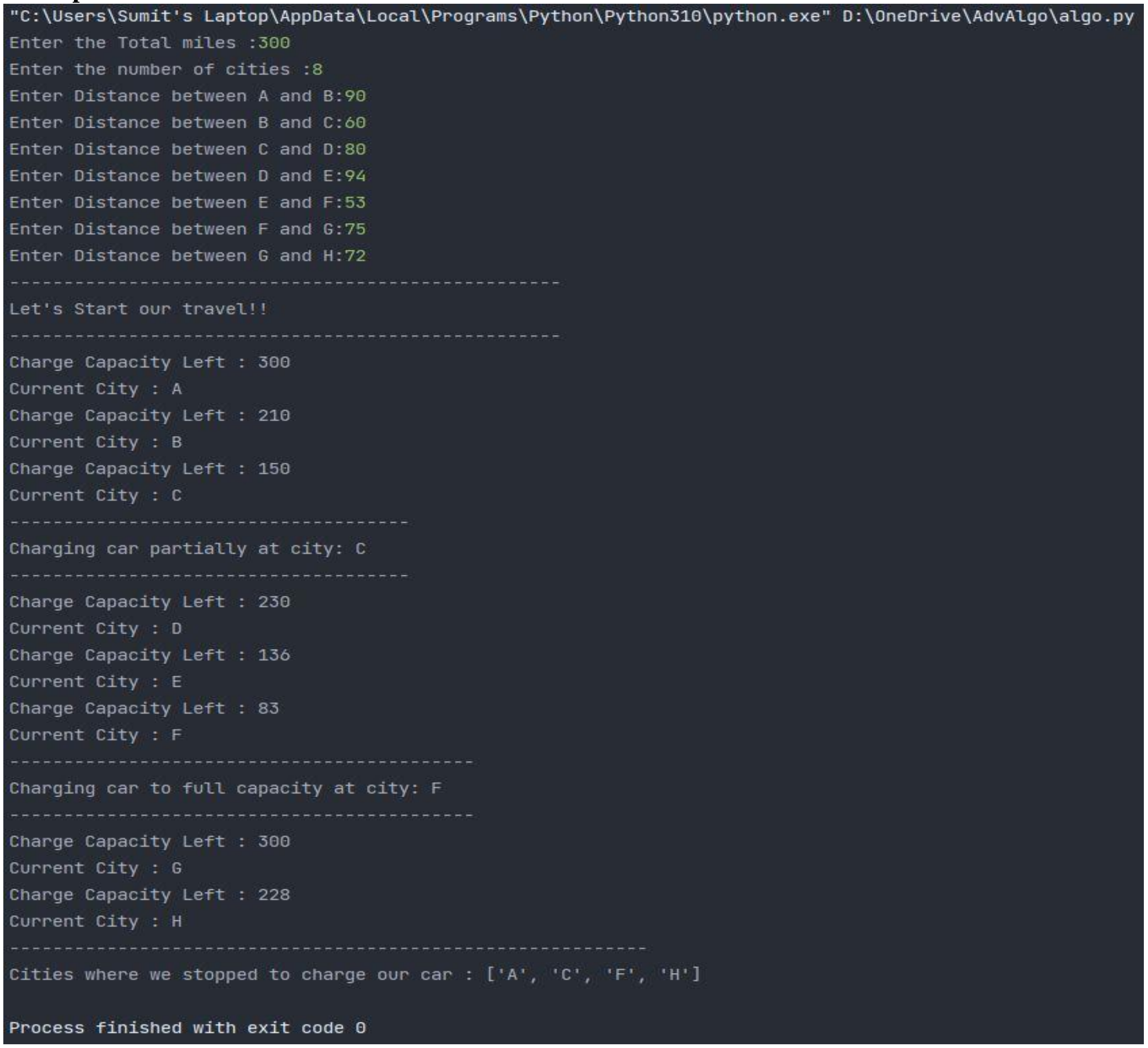
```
"C:\Users\Sumit's Laptop\AppData\Local\Programs\Python\Python310\python.exe" D:\OneDrive\AdvAlgo\algo.py
Enter the Total miles :300
Enter the number of cities :8
Enter Distance between A and B:90
Enter Distance between B and C:94
Enter Distance between C and D:80
Enter Distance between D and E:65
Enter Distance between E and F:83
Enter Distance between F and G:75
Enter Distance between G and H:72
---------------------------------------------------
Let's Start our travel!!
---------------------------------------------------
Charge Capacity Left : 300
Current City : A
Charge Capacity Left : 210
Current City : B
Charge Capacity Left : 116
Current City : C
-------------------------------------
Charging car partially at city: C
-------------------------------------
Charge Capacity Left : 196
Current City : D
Charge Capacity Left : 131
Current City : E
-------------------------------------
Charging car partially at city: E
-------------------------------------
Charge Capacity Left : 214
Current City : F
Charge Capacity Left : 139
Current City : G
-------------------------------------
Charging car partially at city: G
-------------------------------------
Charge Capacity Left : 211
Current City : H
-----------------------------------------------------------
Cities where we stopped to charge our car : ['A', 'C', 'E', 'G', 'H']

Process finished with exit code 0
```

Example 3

Input: The graph below in Figure 1, the capacity is C = 300 miles, the starting city = A and the destination city = H, the miles between cities are shown with blue ink.



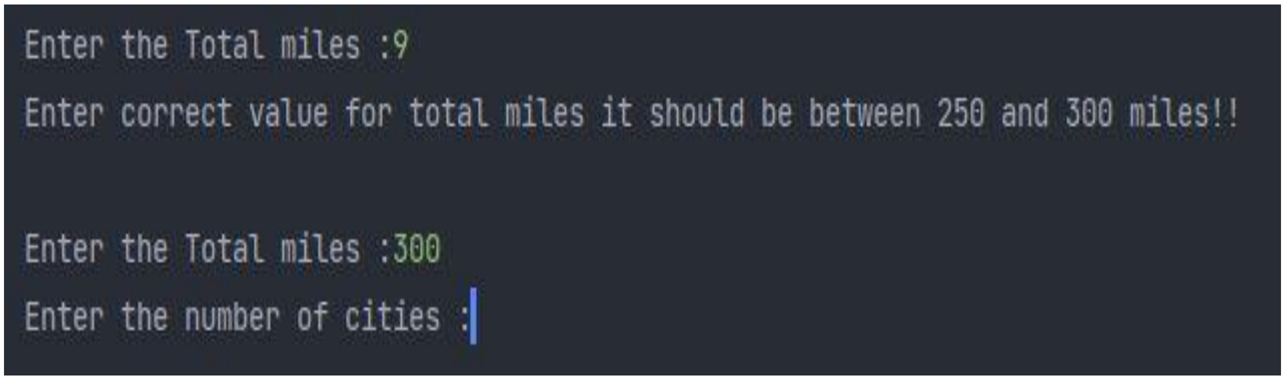
Output:



```
"C:\Users\Sumit's Laptop\AppData\Local\Programs\Python\Python310\python.exe" D:\OneDrive\AdvAlgo\algo.py
Enter the Total miles :300
Enter the number of cities :8
Enter Distance between A and B:90
Enter Distance between B and C:60
Enter Distance between C and D:80
Enter Distance between D and E:94
Enter Distance between E and F:53
Enter Distance between F and G:75
Enter Distance between G and H:72
---------------------------------------------------
Let's Start our travel!!
---------------------------------------------------
Charge Capacity Left : 300
Current City : A
Charge Capacity Left : 210
Current City : B
Charge Capacity Left : 150
Current City : C
-------------------------------------
Charging car partially at city: C
-------------------------------------
Charge Capacity Left : 230
Current City : D
Charge Capacity Left : 136
Current City : E
Charge Capacity Left : 83
Current City : F
-------------------------------------------
Charging car to full capacity at city: F
-------------------------------------------
Charge Capacity Left : 300
Current City : G
Charge Capacity Left : 228
Current City : H
-----------------------------------------------------------
Cities where we stopped to charge our car : ['A', 'C', 'F', 'H']

Process finished with exit code 0
```

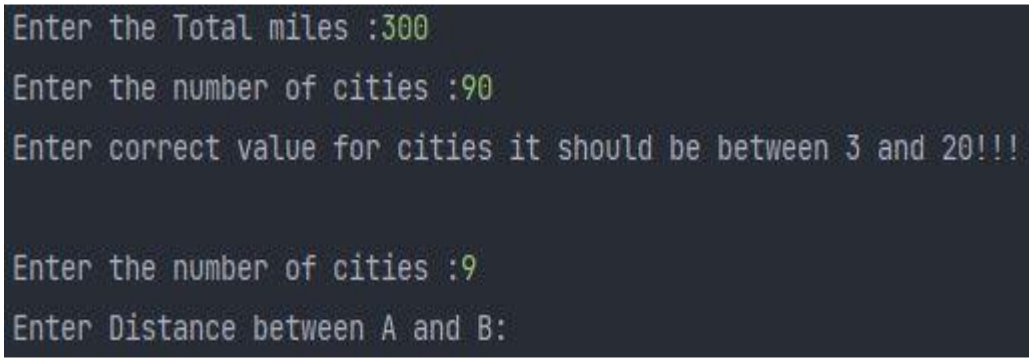
Corner Cases:

1. The capacity *C* in miles is not fixed, but one can assume that is a positive integer between 250 and 350.



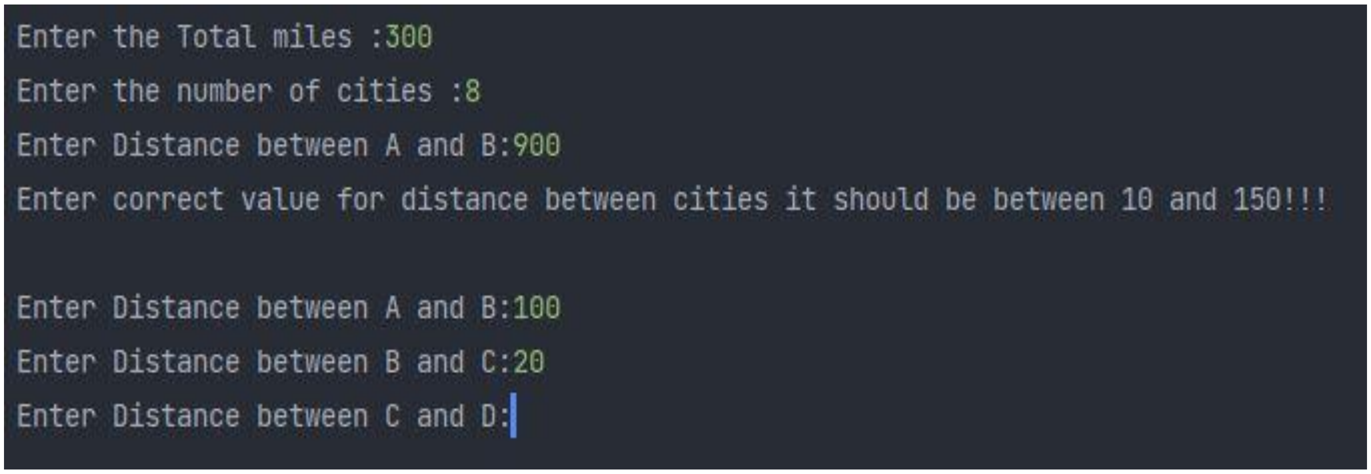
```
Enter the Total miles :9

Enter correct value for total miles it should be between 250 and 300 miles!!


Enter the Total miles :300

Enter the number of cities :
```

2. The number of cities *n* is not fixed, but you can assume that it is greater than 3 and less than 20.



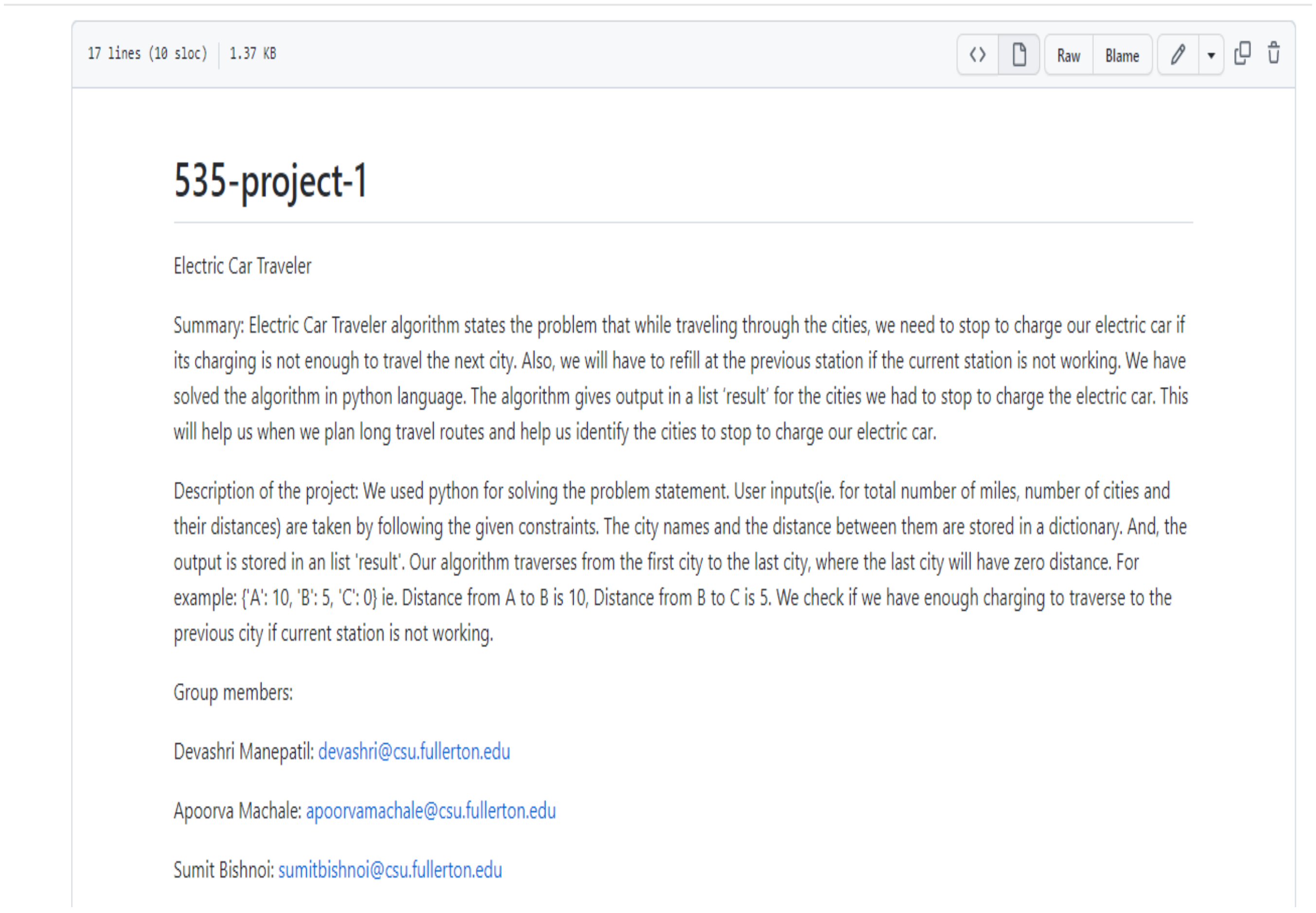
```
Enter the Total miles :300

Enter the number of cities :90

Enter correct value for cities it should be between 3 and 20!!!


Enter the number of cities :9

Enter Distance between A and B:
```

3. The distance between cities is a positive integer and always less than C/2 and greater than 10.



```
Enter the Total miles :300

Enter the number of cities :8

Enter Distance between A and B:900

Enter correct value for distance between cities it should be between 10 and 150!!!


Enter Distance between A and B:100

Enter Distance between B and C:20

Enter Distance between C and D:
```

CHAPTER 6
Group Member Details



535-project-1

Electric Car Traveler

Summary: Electric Car Traveler algorithm states the problem that while traveling through the cities, we need to stop to charge our electric car if its charging is not enough to travel the next city. Also, we will have to refill at the previous station if the current station is not working. We have solved the algorithm in python language. The algorithm gives output in a list 'result' for the cities we had to stop to charge the electric car. This will help us when we plan long travel routes and help us identify the cities to stop to charge our electric car.

Description of the project: We used python for solving the problem statement. User inputs(ie. for total number of miles, number of cities and their distances) are taken by following the given constraints. The city names and the distance between them are stored in a dictionary. And, the output is stored in an list 'result'. Our algorithm traverses from the first city to the last city, where the last city will have zero distance. For example: {'A': 10, 'B': 5, 'C': 0} ie. Distance from A to B is 10, Distance from B to C is 5. We check if we have enough charging to traverse to the previous city if current station is not working.

Group members:

Devashri Manepatil: devashri@csu.fullerton.edu

Apoorva Machale: apoorvamachale@csu.fullerton.edu

Sumit Bishnoi: sumitbishnoi@csu.fullerton.edu