

Submission for Project 1

Introduction:

The algorithm for electric car traveler problem is designed and implemented in the given report. The electric car has capacity C which is the maximum number of miles the electric car can drive. After the capacity is reached the car needs to be recharged. From source to destination, car travels through n cities which have charging stations. There are cases where the chargers may not function in a particular city, in that situation the car should go back to previous station and recharge.

Team Xeno:

Members:

- Hardik Bhawsar (hardik_bhawsar@csu.fullerton.edu)
- Anvit Rajesh Patil (anvit.patil@csu.fullerton.edu)
- Praveen Singh (praveen.singh@csu.fullerton.edu)

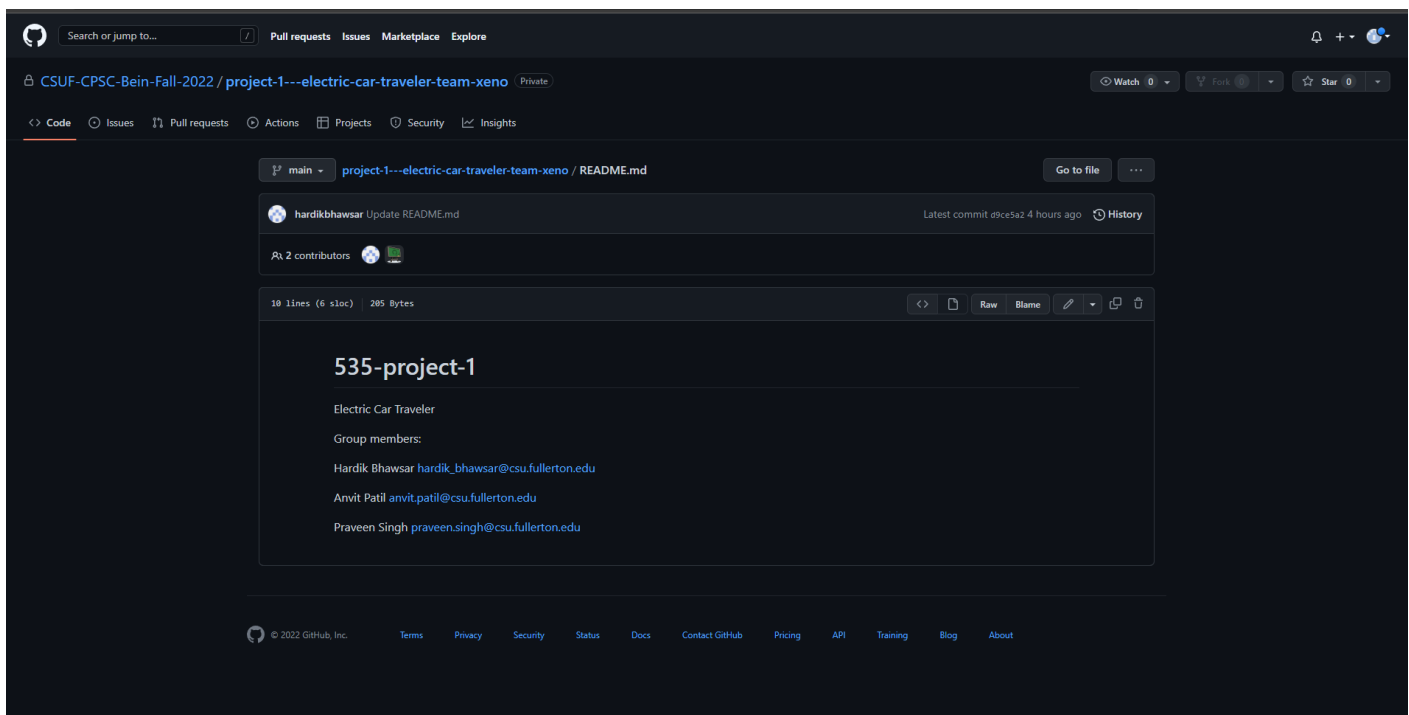


Fig 1: README.md with a list of group members

Assumptions:

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 of 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 can drive to city k and back to the city before city k , in case the charge station in city k is broken.
5. The capacity C in miles is not fixed, but one can assume that is a positive integer between 250 and 350.
6. The number of cities n is not fixed, but you can assume that it is greater than 3 and less than 20.
7. The distance between cities is a positive integer and always less than $C/2$ and greater than 10.

Technologies and tools

C++

For this project we have used C++ as it is a highly portable language, with rich function library. Moreover, it is a powerful, fast and an efficient language making it a first-rate alternative for real-time mathematical simulations like these.

Linked list

We have used Linked list as our choice of data structure for implementing the list of cities. Using it, mitigates the need for a memory overhead whilst providing an excellent access time.

Pseudocode:

- Class Node

- //Node class used to create nodes which contains data and address.

- String name;

- // name of the city

- Int distance;

- //distance of next city from current city

- Node *next;

- //pointer which is pointing to the next city or NULL

- Main()

- //Main function

- Let totalCities, distance, capacity, name;

- //totalCities = number of cities including source and destination

- //distance = distance of next city from the current city

- //capacity = longest distance covered by electric car in a single charge.

- //name = name of the city

- Let *firstCity = NULL

- //*firstCity = head which is pointing to the source city

- Accept the battery capacity of the vehicle as an integer 'capacity' ranging between 250 & 300.

- Cin>>capacity;

- Accept the number of cities as an integer 'totalCities' ranging between 3 and 20.

- Cin>>totalCities;

- For int 'i'=0 to 'totalCities-1'

- Accept City name as a string 'name'.

- Accept distance between the city and the next city as an integer 'distance' which is less than 'capacity/2' and greater than 10.

- Call function addCity(&firstCity, name, distance);

// Function will create a new node with 'name' and 'distance' and will append it in the end of the linked list.

- Call function `chargingCities(firstCity,capacity);`

- `addCity(Node **city, string cityName, int nextCityDistance)`

//Function to create and add new city in the end of the list

//Parameters :

//**city – Pointing to the address of the firstCity/head

//cityName – name of the city which need to be added in the list

//nextCityDistance – Distance of next city from this city

- Create a new Node() with pointer *newCity
 - `Node *newCity = new Node();`
- Create a new pointer which will point to last city , for now point to *city
 - `Node *lastCity = *city`
- Set the name and distance of newCity (new node) with cityName, nextCityDistance
 - `newCity->name = cityName;`
 - `newCity->distance = nextCityDistance;`
 - `newCity->next = NULL;`
- If `*city == NULL`
 - Point the *city to newCity
`*city = newCity;`
- Else
 - Traverse to last node
`While(lastCity-> next != NULL)`
`lastCity = lastCity->next;`
 - Add the new city node after last node
`lastCity->next = newCity;`

- chargingCities(Node *city, int capacity)
 - //Function to find the cities where car need to be recharged.
 - //Parameters:
 - //*city – pointing to the first city(head)
 - //capacity - longest distance covered by electric car in a single charge.
 - Let remainingCapacity = capacity
 - While city->next is not NULL //Traverse till last node
 - If 'remainingCapacity' is greater than 2* city->distance
 - remainingCapacity= remainingCapacity – city->distance
 - city=city->next
 - Else //print city where car needs to recharge
 - print (city->name)
 - remainingCapacity= capacity
 - Finally print the last city name.

Implementation:

- Open the terminal.
- Change the directory to access the required code (~/project-1---electric-car-traveler-team-xeno in our case). Using 'cd' command. (\$ cd project-1---electric-car-traveler-team-xeno).
- Compile the code using (\$ g++ code.cpp).
- Run the executable using (\$./a.out)

Working:

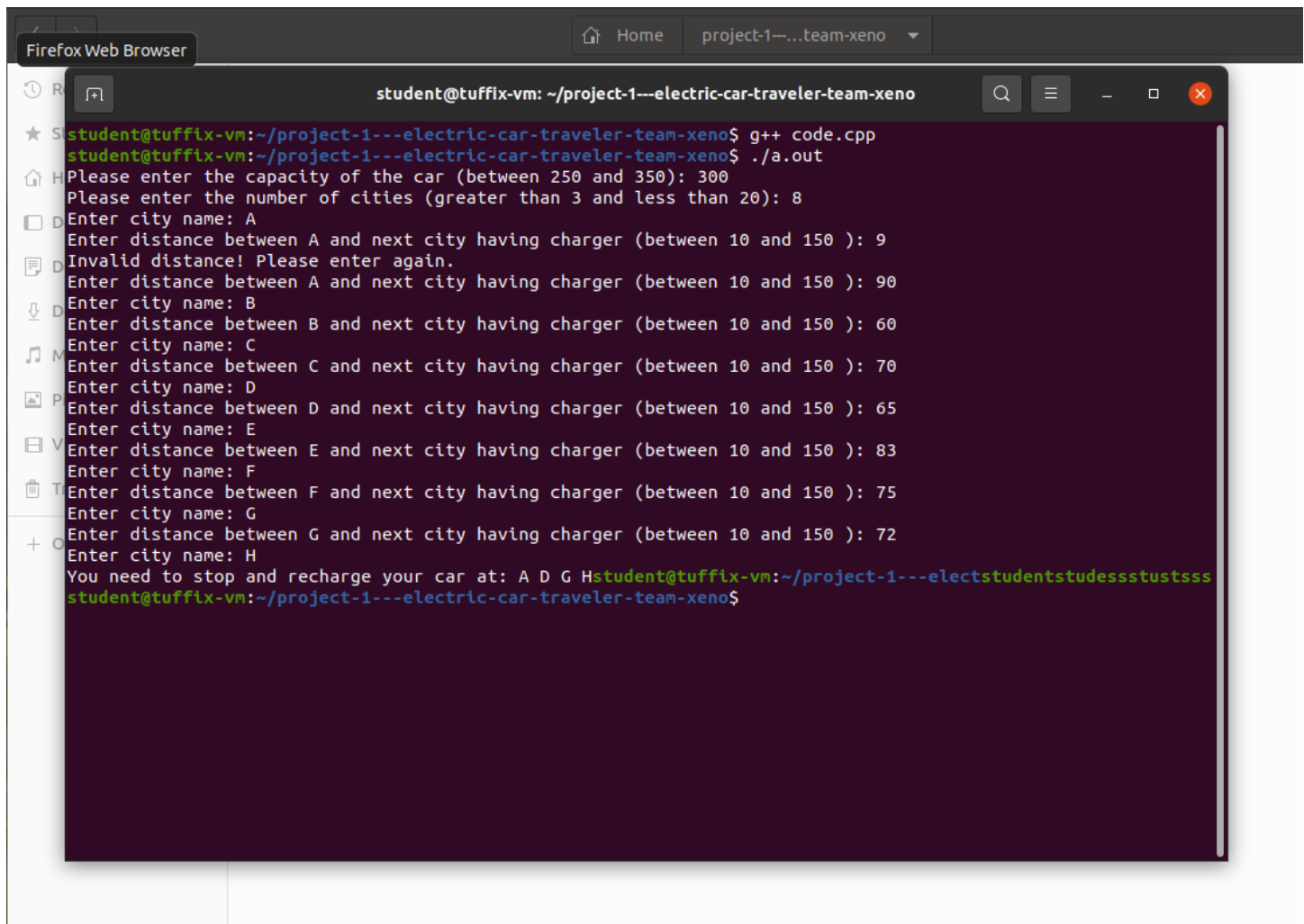
Example 1:

Input:

A -90-> B -60-> C -70-> D -65-> E -83-> F -75-> G -72-> H

Output:

A, D, G, H.



```
student@tuffix-vm: ~/project-1---electric-car-traveler-team-xeno
$ g++ code.cpp
student@tuffix-vm:~/project-1---electric-car-traveler-team-xeno$ ./a.out
Please enter the capacity of the car (between 250 and 350): 300
Please enter the number of cities (greater than 3 and less than 20): 8
Enter city name: A
Enter distance between A and next city having charger (between 10 and 150 ): 9
Invalid distance! Please enter again.
Enter distance between A and next city having charger (between 10 and 150 ): 90
Enter city name: B
Enter distance between B and next city having charger (between 10 and 150 ): 60
Enter city name: C
Enter distance between C and next city having charger (between 10 and 150 ): 70
Enter city name: D
Enter distance between D and next city having charger (between 10 and 150 ): 65
Enter city name: E
Enter distance between E and next city having charger (between 10 and 150 ): 83
Enter city name: F
Enter distance between F and next city having charger (between 10 and 150 ): 75
Enter city name: G
Enter distance between G and next city having charger (between 10 and 150 ): 72
Enter city name: H
You need to stop and recharge your car at: A D G H
student@tuffix-vm:~/project-1---electric-car-traveler-team-xeno$
```

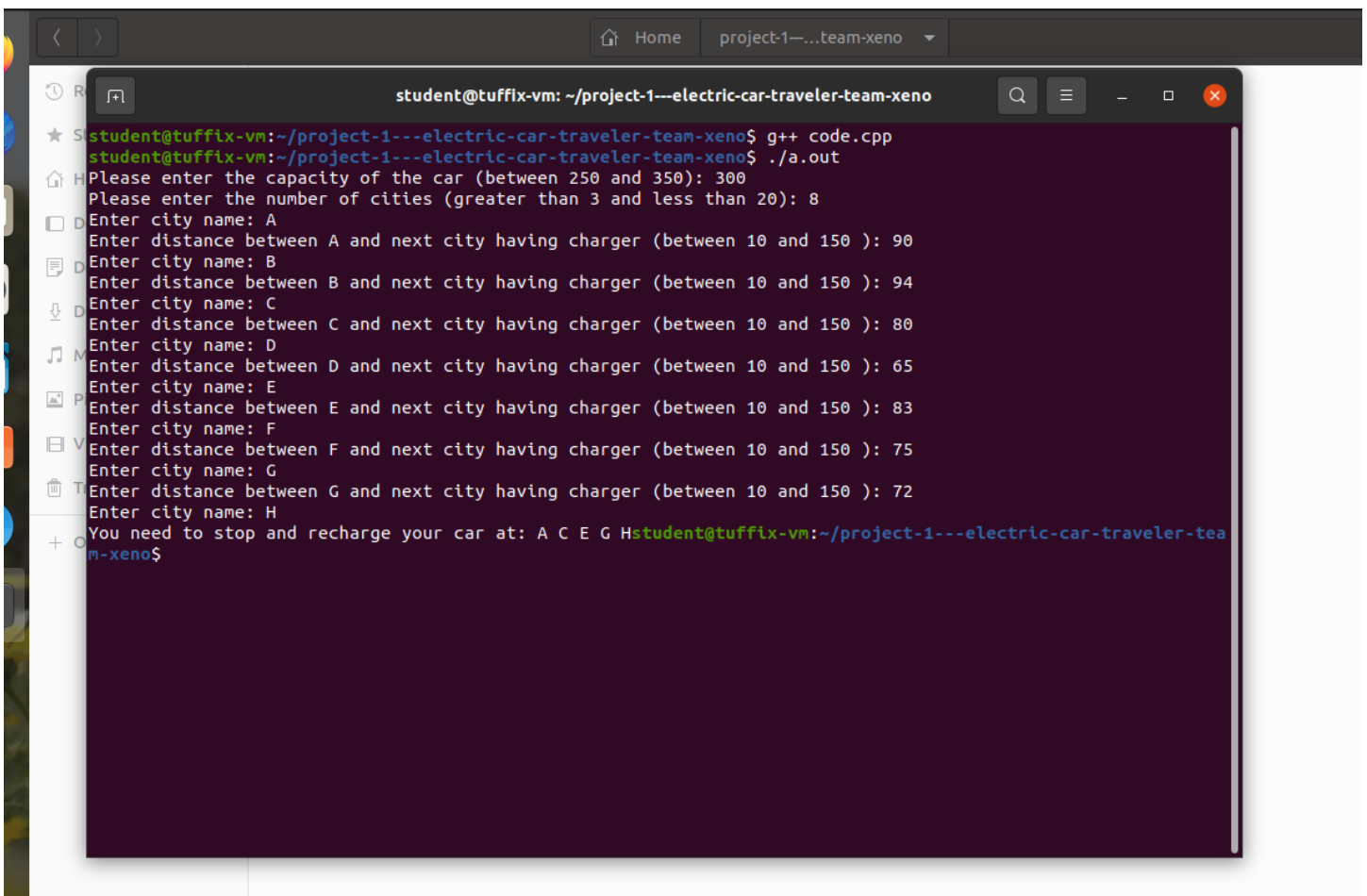
Example 2:

Input:

A -90-> B -94-> C -80-> D -65-> E -83-> F -75-> G -72-> H

Output:

A, C, E, G, H.



```
student@tuffix-vm: ~/project-1---electric-car-traveler-team-xeno
student@tuffix-vm:~/project-1---electric-car-traveler-team-xeno$ g++ code.cpp
student@tuffix-vm:~/project-1---electric-car-traveler-team-xeno$ ./a.out
Please enter the capacity of the car (between 250 and 350): 300
Please enter the number of cities (greater than 3 and less than 20): 8
Enter city name: A
Enter distance between A and next city having charger (between 10 and 150 ): 90
Enter city name: B
Enter distance between B and next city having charger (between 10 and 150 ): 94
Enter city name: C
Enter distance between C and next city having charger (between 10 and 150 ): 80
Enter city name: D
Enter distance between D and next city having charger (between 10 and 150 ): 65
Enter city name: E
Enter distance between E and next city having charger (between 10 and 150 ): 83
Enter city name: F
Enter distance between F and next city having charger (between 10 and 150 ): 75
Enter city name: G
Enter distance between G and next city having charger (between 10 and 150 ): 72
Enter city name: H
You need to stop and recharge your car at: A C E G H
student@tuffix-vm:~/project-1---electric-car-traveler-team-xeno$
```

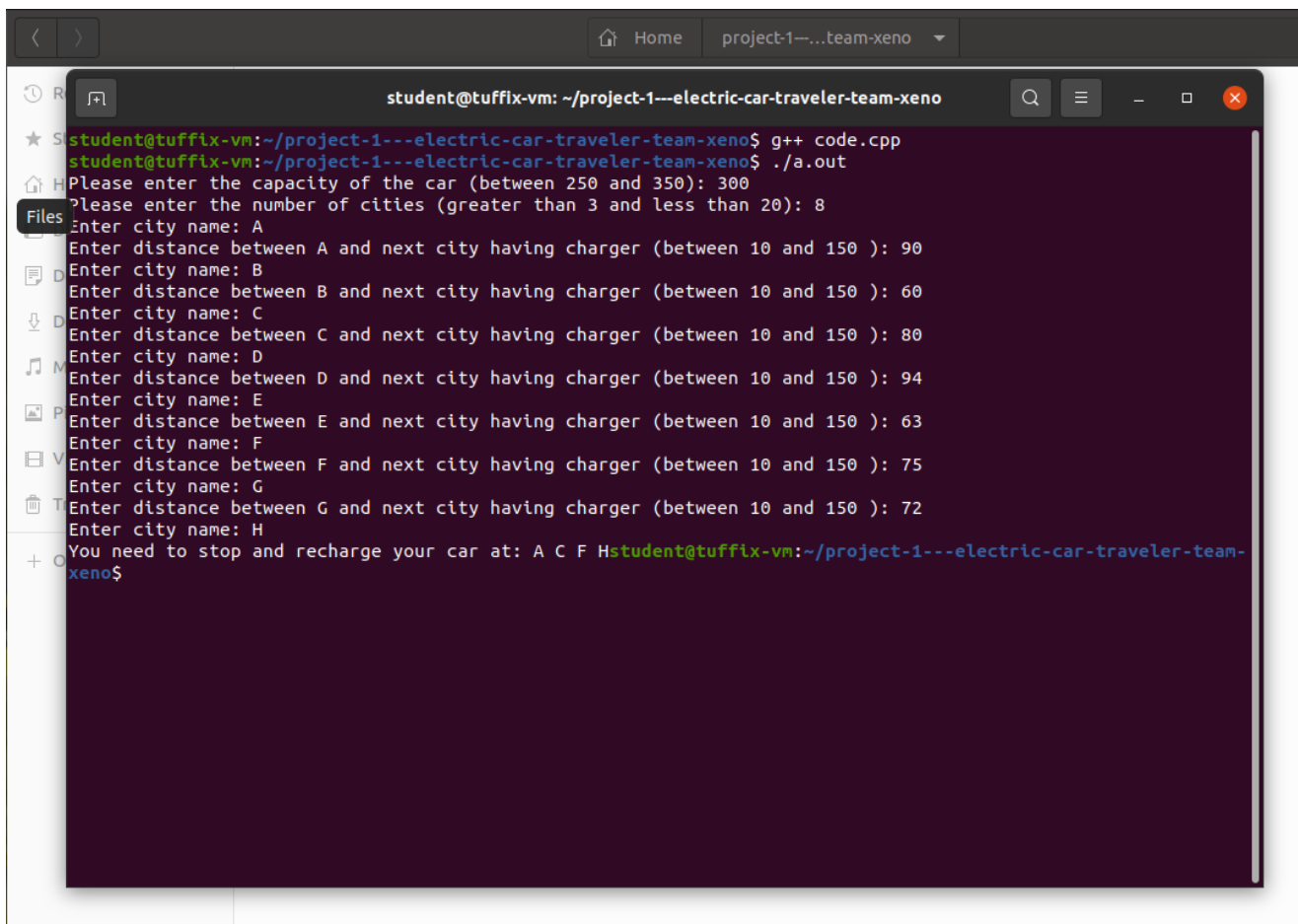
Example 3:

Input:

A -90-> B -60-> C -80-> D -94-> E -53-> F -75-> G -72-> H

Output:

A, C, F, H.



```
student@tuffix-vm: ~/project-1---electric-car-traveler-team-xeno
student@tuffix-vm:~/project-1---electric-car-traveler-team-xeno$ g++ code.cpp
student@tuffix-vm:~/project-1---electric-car-traveler-team-xeno$ ./a.out
Please enter the capacity of the car (between 250 and 350): 300
Please enter the number of cities (greater than 3 and less than 20): 8
Enter city name: A
Enter distance between A and next city having charger (between 10 and 150 ): 90
Enter city name: B
Enter distance between B and next city having charger (between 10 and 150 ): 60
Enter city name: C
Enter distance between C and next city having charger (between 10 and 150 ): 80
Enter city name: D
Enter distance between D and next city having charger (between 10 and 150 ): 94
Enter city name: E
Enter distance between E and next city having charger (between 10 and 150 ): 63
Enter city name: F
Enter distance between F and next city having charger (between 10 and 150 ): 75
Enter city name: G
Enter distance between G and next city having charger (between 10 and 150 ): 72
Enter city name: H
You need to stop and recharge your car at: A C F H
student@tuffix-vm:~/project-1---electric-car-traveler-team-xeno$
```


Summary:

In the report, the algorithm for electric car traveler problem is designed and implemented. In the implementation, linked list data structure is used for easy traversal between different cities. Each city is represented with a node having data as name of the city and distance of next city from the current city. Every city is inserted in the end of the list. List of cities where the car needs to be recharged is also displayed in the end. There are also few assumptions as mentioned above in report which needs to be considered for proper execution of the code. Code is tested with three example and picture of their output is also pasted above in report.

Reference

[1] CPSC 535 Fall 2022 Project 1

<https://docs.google.com/document/d/1xqOCSsTEQKHKjnFkbDd9nAlihBM7qCSaZ0dE6-OwUgk>