Ben Wedeen, Captain Payne, Connor Ely

Professor Gupta

Data Structures

22 April 2019

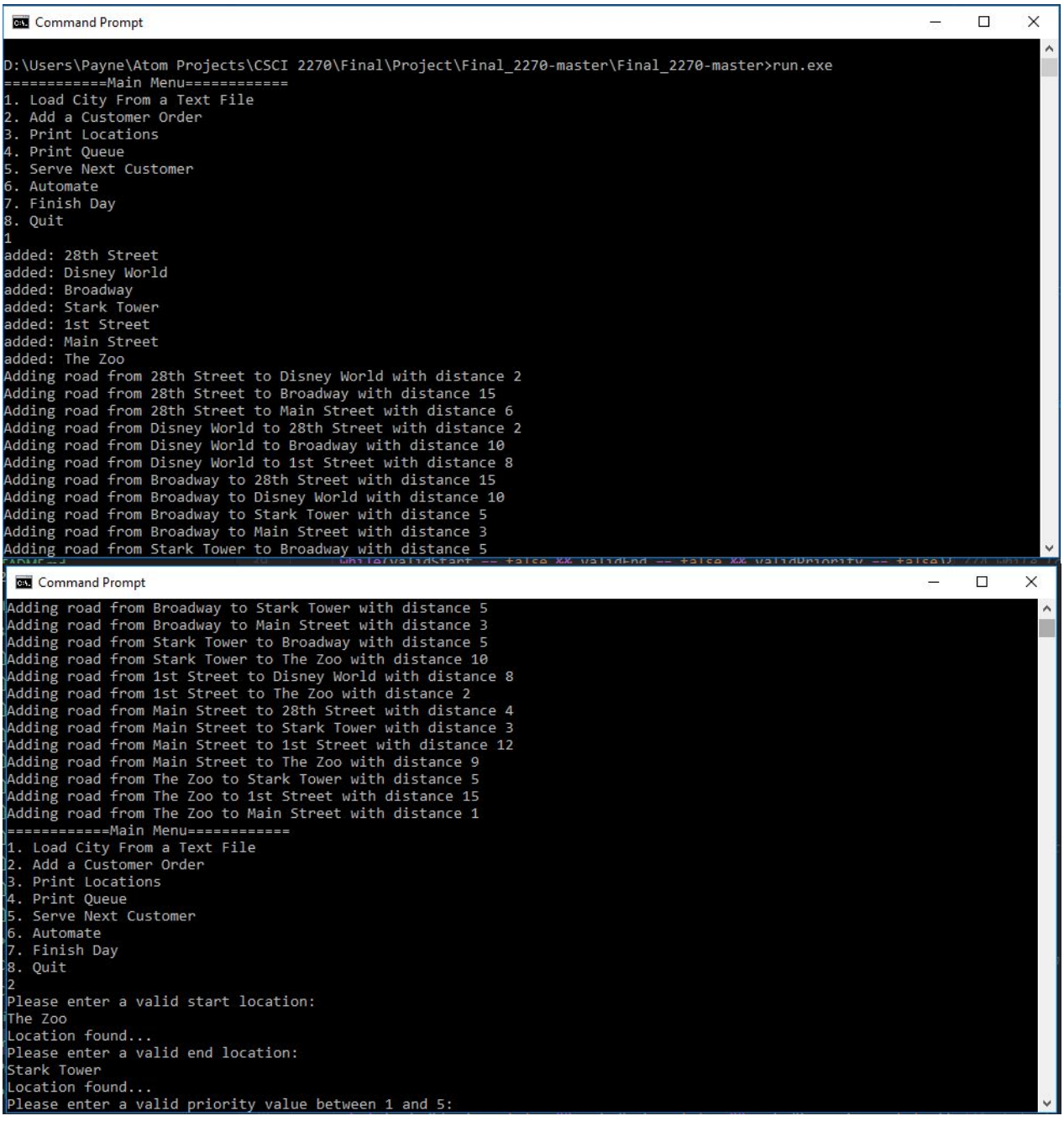
Self Driving Electric Car System

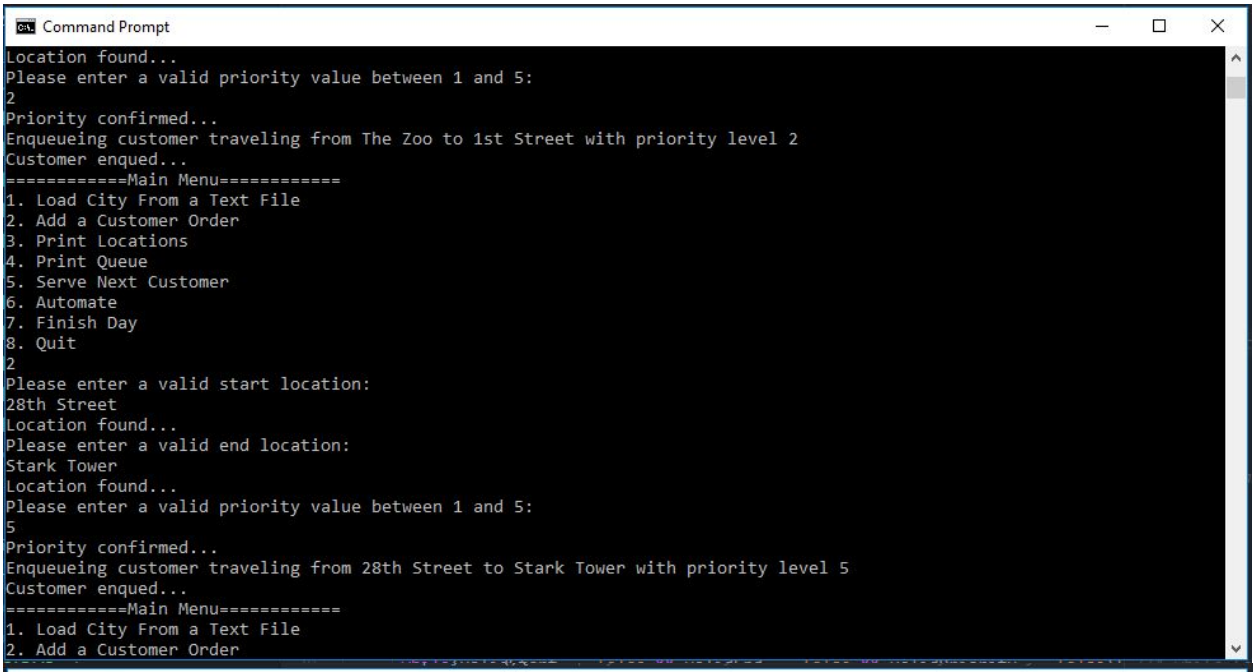
Our project mainly utilized graphs and heap data structures to accomplish our goal. Our graph was used to properly map out a basic city, with various locations represented by vertices and the roads connecting them represented by edges. The heap structure acted as a priority queue, allowing us to selectively prioritize customers seeking to travel from set start and finish locations based upon both the distance they needed to traverse across the city as well as their priority level. Our project works by providing the user with a menu of eight options. The first option reads in a file, "cityFile.txt", which contains locations as well as numbers representing the roads connecting them. Using the functions of the graph data structure, it constructs a basic layout for a simple city. The second option involves adding customers to the priority queue, giving the user an option to input start and finish locations, as well as a priority level between one and five. The third prints out a list of locations, with the option to traverse using either breadth first traversal or depth first traversal and the fourth option outputs the current customers in the queue.

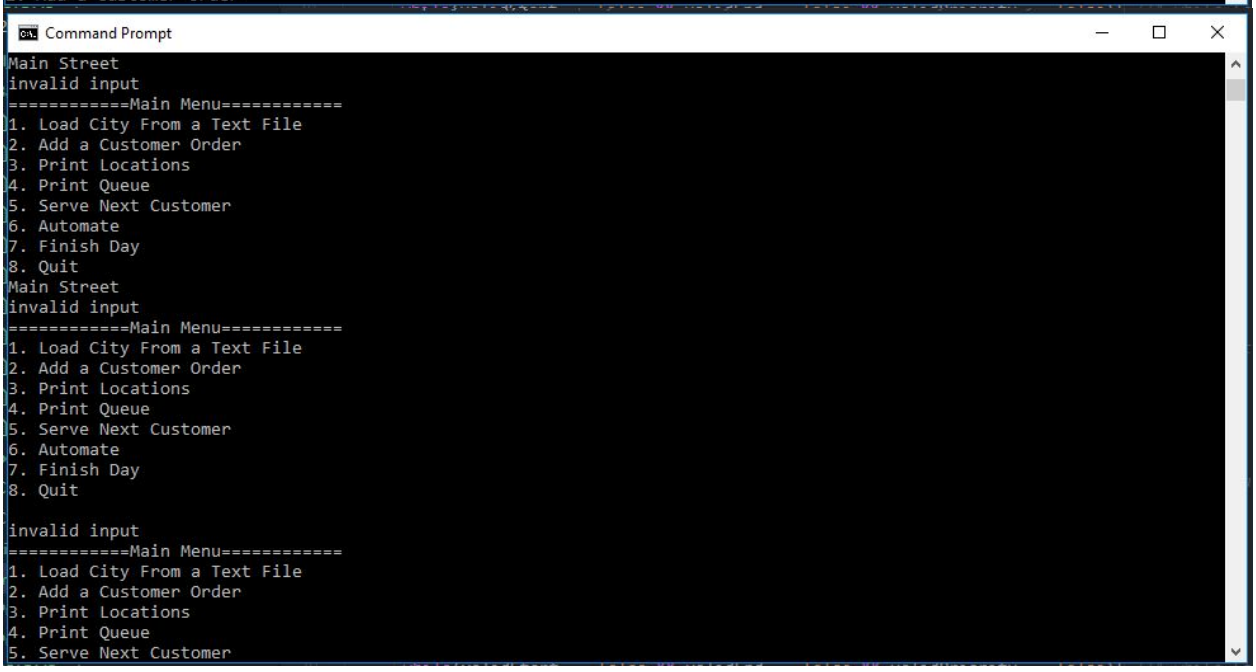
The fifth option serves the next customer in the priority queue, and calculates the cost of the trip based on the distance between the customer's start and end locations (using a modified version of Dijkstra's algorithm) as well as their priority level. If the car transporting the customer is low on charge, a small sum is reduced from that customer's fee to make up for the amount of

time needed to wait for the car to recharge. Once the current customer's job is completed, they are dequeued from the priority queue. The sixth option acts as an automated version of the fifth option, running through the priority queue until every customer has been successfully served. The seventh option finishes the current day, and stores the amount of profit in a .txt file. The eighth and final option ends the program.

Below are pictures of some of the outputs of our program:




Command Prompt
Location found...
Please enter a valid priority value between 1 and 5:
2
Priority confirmed...
Enqueueing customer traveling from The Zoo to 1st Street with priority level 2
Customer enqued...
==========Main Menu==========
1. Load City From a Text File
2. Add a Customer Order
3. Print Locations
4. Print Queue
5. Serve Next Customer
6. Automate
7. Finish Day
8. Quit
2
Please enter a valid start location:
28th Street
Location found...
Please enter a valid end location:
Stark Tower
Location found...
Please enter a valid priority value between 1 and 5:
5
Priority confirmed...
Enqueueing customer traveling from 28th Street to Stark Tower with priority level 5
Customer enqued...
==========Main Menu==========
1. Load City From a Text File
2. Add a Customer Order


Command Prompt
Main Street
invalid input
==========Main Menu==========
1. Load City From a Text File
2. Add a Customer Order
3. Print Locations
4. Print Queue
5. Serve Next Customer
6. Automate
7. Finish Day
8. Quit
Main Street
invalid input
==========Main Menu==========
1. Load City From a Text File
2. Add a Customer Order
3. Print Locations
4. Print Queue
5. Serve Next Customer
6. Automate
7. Finish Day
8. Quit
invalid input
==========Main Menu==========
1. Load City From a Text File
2. Add a Customer Order
3. Print Locations
4. Print Queue
5. Serve Next Customer