***Section 1: Programming/Coding***

A.  Identify the algorithm that will be used to create a program to deliver the packages and meets all requirements specified in the scenario.

**-Greedy Algorithm**

B.  Write a core algorithm overview, using the sample given, in which you do the following:

1.  Comment using pseudocode to show the logic of the algorithm applied to this software solution.  
**- Create a list of locations from the corresponding CSV file**

**- Create a list of distances from the corresponding CSV file**

**- Load presorted packages onto trucks, then load priority packages, then load all remaining packages**

**- While delivering, first, deliver packages with the highest priority. If packages have the same priority level deliver the closest one.**

**- Repeat until done.**

2.  Apply programming models to the scenario.

-**The main program creates a hash table of packages that are pulled from a CSV file that contains data about all the packages. Then a package manager is created to help set package priority and simulation running. After the simulation is done running, the manager will show a user interface where the user can pull certain information about packages and their statuses at specific times.**

3.  Evaluate space-time complexity using Big O notation throughout the coding and for the entire program.

\***Commented throughout program\***

4.  Discuss the ability of your solution to adapt to a changing market and scalability.

**The solution can adapt to changes in the market by simply adding the extra nodes for distances and locations. The program may slow down for larger data sets and higher package count.**

5.  Discuss the efficiency and maintainability of the software.

**-The efficiency of the software is good for decent-sized uses. However, when we start loading excessive data into the program, it will slow down and become prone to errors since there are a lot of assumptions taking place.**

**-The maintainability of the software isn’t the best. If we want to add more data to the process we can do that by simply adding it to the CSV files. However; some packages are coded directly into the truck. To remove or add these specific packages we will need to edit the code where they are assigned.**

6.  Discuss the self-adjusting data structures chosen and their strengths and weaknesses based on the scenario.

**The self-adjusting data structure chosen was the list of packages inside the truck. The strength is that when we deliver a package, we can simply pop it out of the packages loaded list. However, since we have priorities set up, we will have to traverse the list and find all the highest priority packages first, then traverse through the packages to find the closest one. The runtime when accessed isn’t the best. Since we are typically going through the list numerous times we end up with a complexity of O(N) up to O(N^2).**

C.  Write an original code to solve and to meet the requirements of the lowest mileage usage and having all packages delivered on time.

* **Create a comment within the first line of your code that includes your first name, last name, and student ID.**
* **Include comments at each block of code to explain the process and flow of the coding.**

D.  Identify a data structure that can be used with your chosen algorithm to store the package data.

**A hash table was used to store information (including priority levels) about all packages, but a list was used to store packages loaded on a truck.**

1. Explain how your data structure includes the relationship between the data points you are storing.

**The relationship between the data points is the package’s ID. Through the ID we can locate packages inside the hash table and the truck’s list.**

E.  Develop a hash table, without using any additional libraries or classes, with an insertion function that takes the following components as input and inserts the components into the hash table:

•  package ID number

•  delivery address

•  delivery deadline

•  delivery city

•  delivery zip code

•  package weight

•  delivery status (e.g., delivered, in route)

F.  Develop a lookup function that takes the following components as input and returns the corresponding data elements:

•  package ID number

•  delivery address

•  delivery deadline

•  delivery city

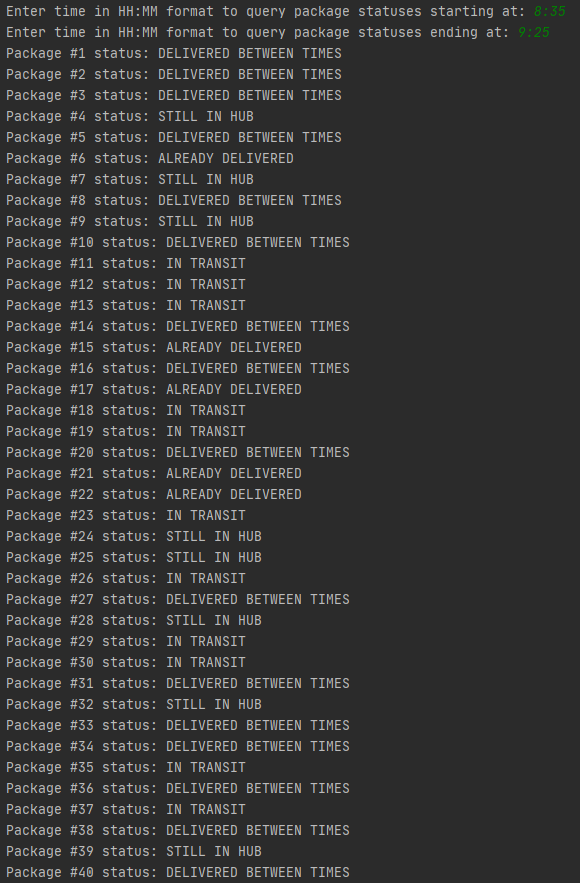
•  delivery zip code

•  package weight

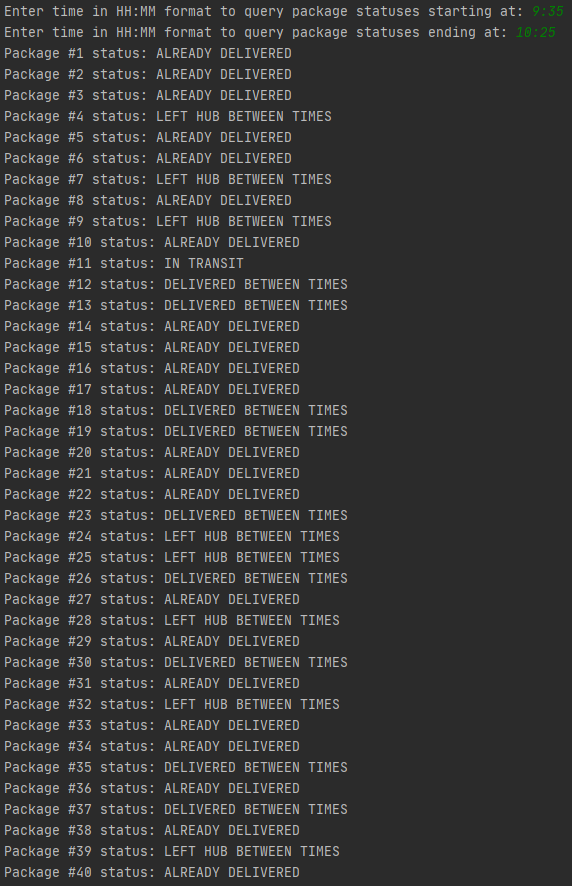
•  delivery status (e.g., delivered, in route)

G.  Provide an interface for the insert and lookup functions to view the status of any package at any time. This function should return all information about each package, including delivery status.

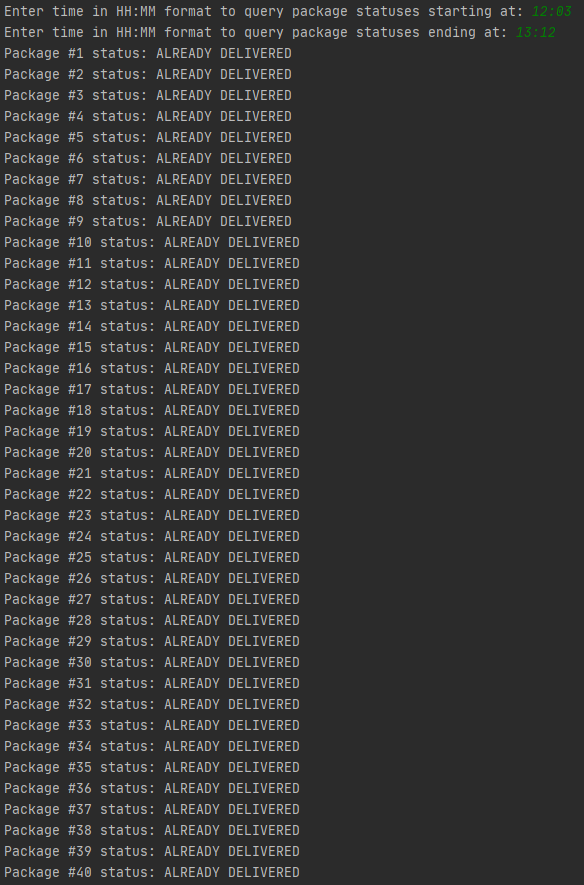
1.  Provide screenshots to show the package status of all packages at a time between 8:35 a.m. and 9:25 a.m.

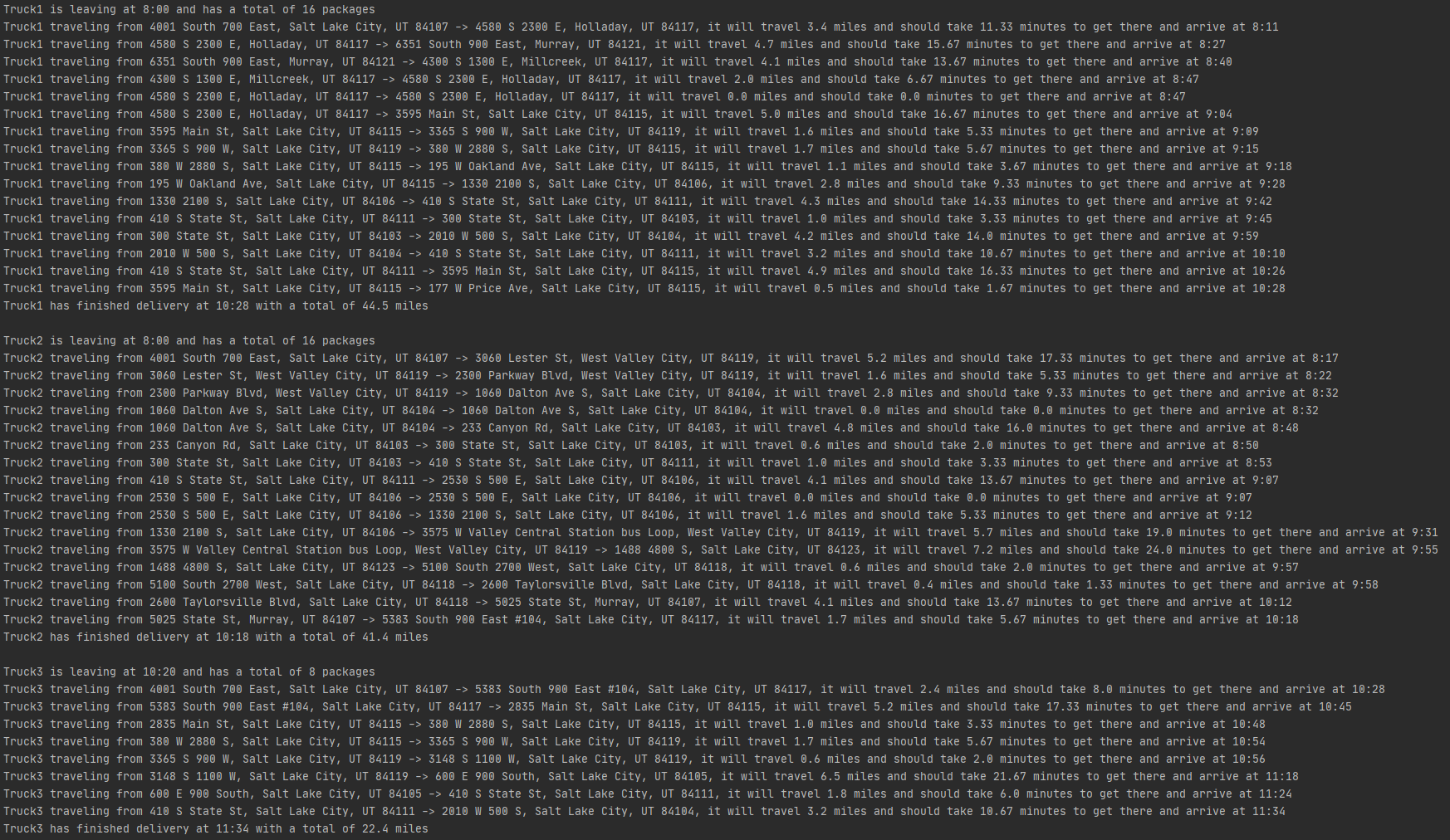


1. Provide screenshots to show the package status of all packages at a time between 9:35 a.m. and 10:25 a.m.



1. Provide screenshots to show the package status of all packages at a time between 12:03 p.m. and 1:12 p.m.



H.  Run your code and provide screenshots to capture the complete execution of your code.

.

.

******.

***Section 2: Annotations***

I.  Justify your choice of the algorithm by doing the following: **Nearest Neighbor**

1.  Describe at least **two** strengths of the algorithm you chose.

**- Easy to understand and implement**

**- Simple enough to debug without too much of a hassle**

2.  Verify that the algorithm you chose meets all the criteria and requirements given in the scenario.

* **Total miles 108.3 is under the 140-mile limit whilst delivering packages on time.**

1. Identify **two** other algorithms that could be used and would have met the criteria and requirements given in the scenario.

**- Dijkstra’s Algorithm**

**- Ant Colony Optimization Algorithm**

a.  Describe how each algorithm identified in part I3 is different from the algorithm you chose to use in the solution.

**- Dijkstra’s Algorithm can be used to determine routes before leaving the hub making it possible to assign packages near each other, to begin with. The Nearest Neighbor algorithm is more ‘On the go’.**

-**Ant Colony Optimization Algorithm is used to find the shortest path in a graph to a point or in our case a prioritized package. Since we are traveling towards that package, we can find packages that are on the way to not waste miles are driven.**

J.  Describe what you would do differently if you did this project again.

**- I would set up a little differently where I would group packages before I start routing them out. This would help inefficiency in delivering packages and minimize the miles driven. I would also load prioritized packages differently so that one truck won’t get all prioritized packages. This will be a problem when more packages have deadlines in the future. I should load prioritized packages onto trucks by turns so they won’t all stick onto one but instead will be evenly distributed. Another option would be creating a queue of prioritized packages.**

K.  Justify your choice of data structure by doing the following:

1.  Verify that the data structure you chose meets all the criteria and requirements given in the scenario.

* **Hash and lookup functions work whilst giving the ability to deliver on time and have the least number of miles.**

1. Describe the efficiency of the data structure chosen.

**- A list of prioritized packages in a truck helps sort through what needs to be delivered first letting us deliver by set deadlines. This data structure is used to find what package to deliver next. Since we have to iterate over all the packages, we have a time complexity of O(N).**

**- A hash table of all packages is used to store and retrieve data through a package’s ID. Using a hash table removes the need to go through all the data elements stored inside the table. This helps us achieve a complexity of O(1) by directly accessing a data element based on its ID.**

b.  Explain the expected overhead when linking to the next data item.

**-An overhead is any extra time, memory, or resources required to perform a task. The expected overhead when linking to the next data item is memory and time. We will take an extra step (time) to link one data item to the next while also taking extra space (memory) to store that information.**

c.  Describe the implications of when more package data is added to the system or other changes in scale occur.

**-If we were to upscale our data, going through the lists would become slower as the data grows. This will cause our overall program to slow down since it must traverse through every package numerous times. Also, our trucks have a limit of 16 packages. They would have to return at least once more and load more packages.**

2.  Identify **two** other data structures that can meet the same criteria and requirements given in the scenario.

**- Queues**

**- Stacks**

a.  Describe how each data structure identified in part K2 is different from the data structure you chose to use in the solution.

**- Queues can help when dealing with priorities. Queues would make more sense since you can load higher priority objects first having them pop. This is helpful because we would not have to traverse through our list every time**

**-We can also use stacks where we put the furthest object with the least priority in first until we hit the closest object with the highest priority. This is different than a simple list because we do not have to traverse through the list every time.**

L.   Acknowledge sources, using in-text citations and references, for content that is quoted, paraphrased or summarized.

M.  Demonstrate professional communication in the content and presentation of your submission.