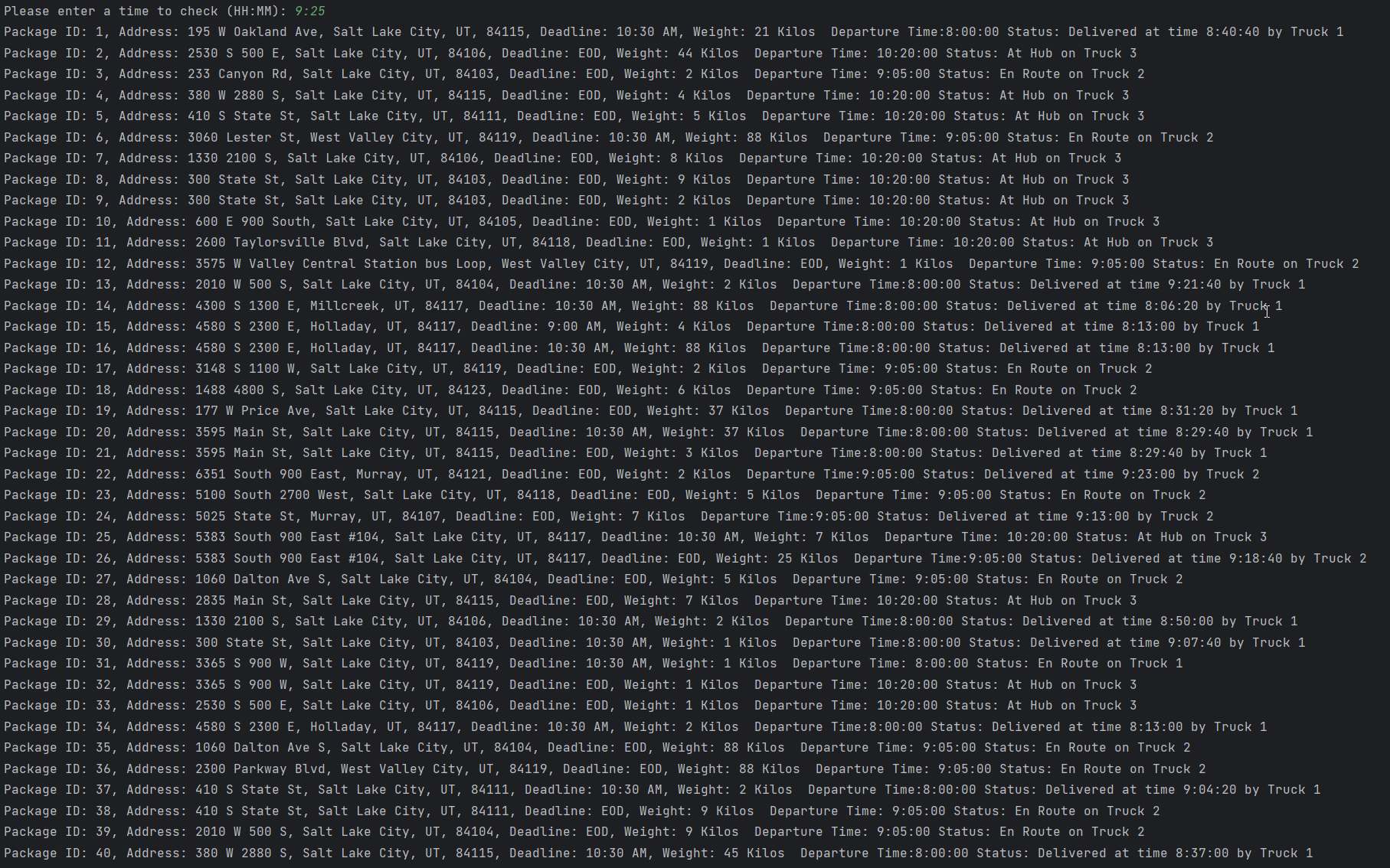
**Task 2 Written Submission**

**By: Michael Vu  
Student ID: 010845718**

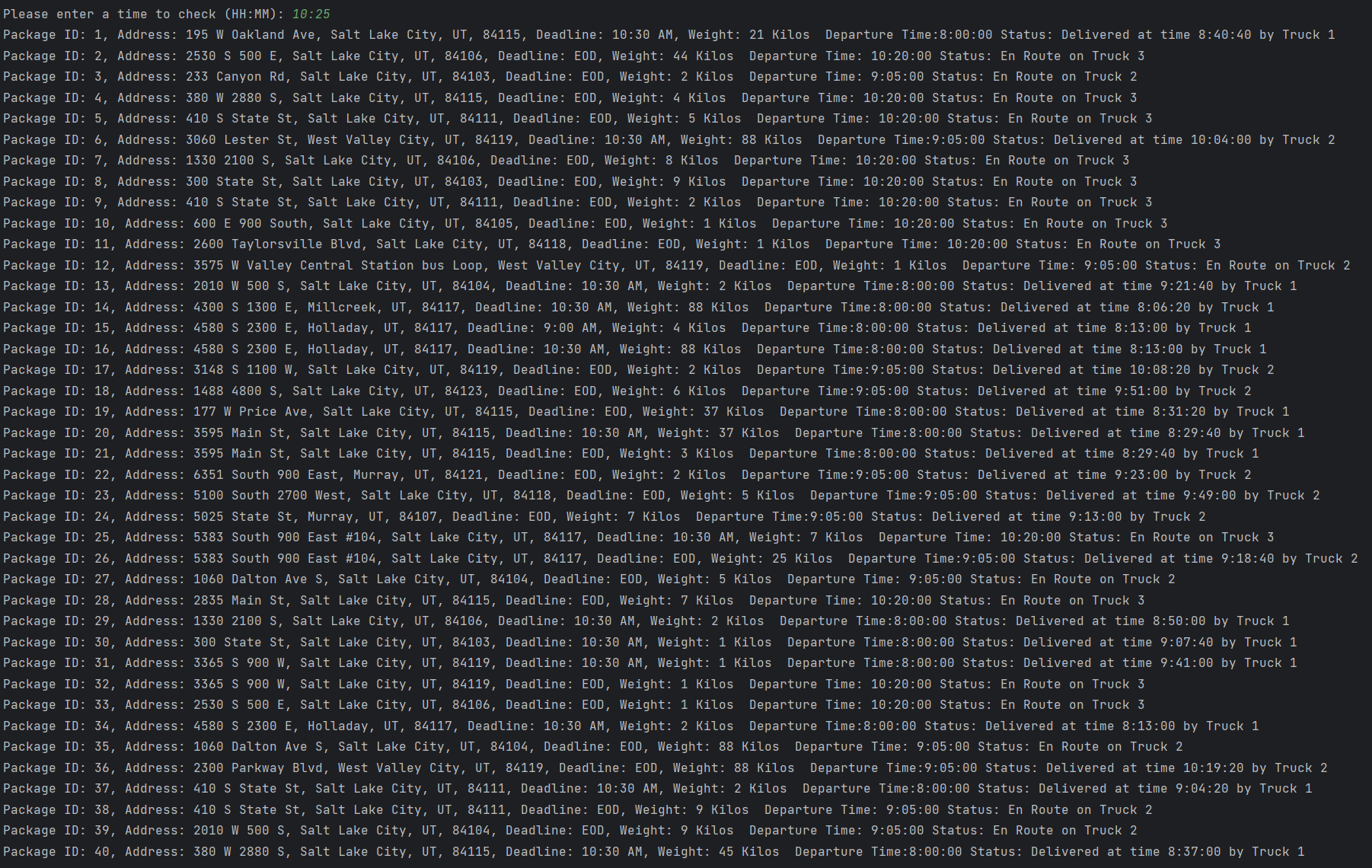
**Date: October 9, 2023**

**D. Provide an intuitive interface for the user to view the delivery status (including the delivery time) of any package at any time and the total mileage traveled by all trucks.**

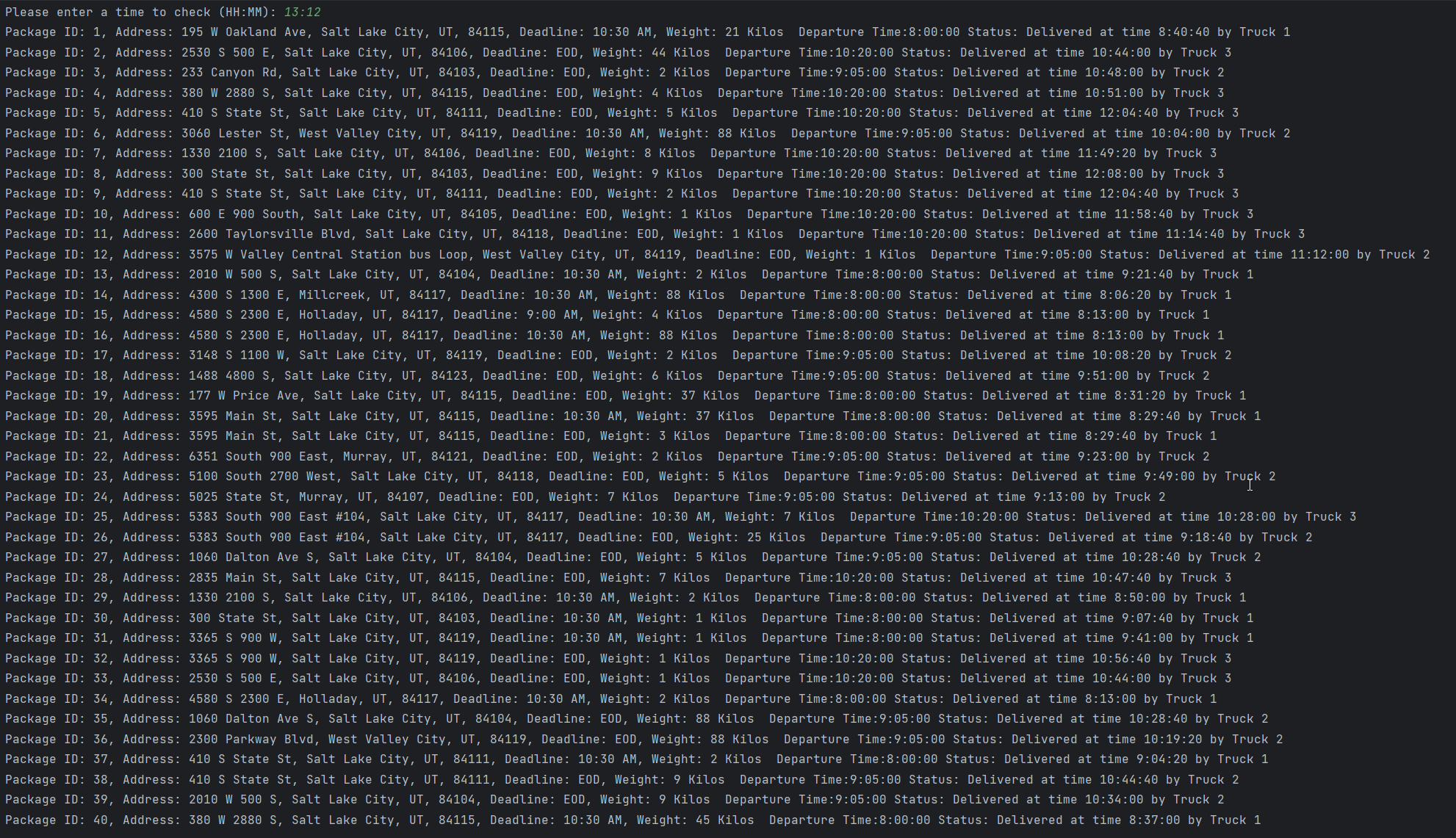
**1. Provide screenshots to show the status of all packages loaded onto each truck at a time between 8:35 a.m. and 9:25 a.m.**



**2. Provide screenshots to show the status of all packages loaded onto each truck at a time between 9:35 a.m. and 10:25 a.m.**



**3. Provide screenshots to show the status of all packages loaded onto each truck at a time between 12:03 p.m. and 1:12 p.m.**



**E. Provide screenshots showing successful completion of the code that includes the total mileage traveled by all trucks.**



**F. Justify the package delivery algorithm used in the solution as written in the original program by doing the following:**

1. **Describe two or more strengths of the algorithm used in the solution.**

The strengths of the nearest neighbor algorithm are firstly, it is very easy to implement and understand, it does not require complex mathematical equations or algorithms, making it accessible for beginners and non-experts. Secondly, it provides a faster solution compared to other algorithms, due to the

simplicity of the algorithm.

1. **Verify that the algorithm used in the solution meets all requirements in the scenario.**

The nearest neighbor algorithm meets all the requirements in the scenario; it delivers all the packages within 140 miles, all packages are delivered within the specified deadlines, and also accommodates for the packages with special notes, such as packages that have to be on truck 2.

**3. Identify two other named algorithms that are different from the algorithm implemented in the solution and would meet all requirements in the scenario.**

Two other algorithms that could be used in this scenario are Djikstra’s Algorithm and the Depth-First Search algorithm.

**a. Describe how both algorithms identified in part F3 are different from the algorithm used in the solution.**

The nearest neighbor algorithm is quite short-sighted­­­– it chooses the nearest node (or location) to its current node, without trying to optimize the route past its next target–while on the other hand, Dijkstra’s algorithm aims to find the shortest path from one specific node to every other node on a graph. It does this by selecting the node with the smallest distance, explores its neighbors, and updates the optimal route if a shorter path is found to the target node.

The Depth-First Search (DFS) algorithm is a much more complex and time-consuming solution compared to the nearest neighbor algorithm. Given a set of nodes to be traversed, DFS will iterate through every single combination possible that results in all the nodes being visited. The disadvantage of this algorithm is its computation time of *n!*, but the advantage is that it will find the most optimal solution.

**G. Describe what you would do differently, other than the two algorithms identified in part F3, if you did this project again, including details of the modifications that would be made.**

If I were to do this project again, instead of manually adding a list of packages to each truck object, I would create a method with the purpose of loading the packages onto the trucks based on the special notes (such as package #18 having to be on truck 2). This could possibly be done by extracting the string in the special notes column, and implementing if/elif/else statements to handle the various requirements. This method would improve the functionality and performance of the program as the number of packages increases. Additionally, I would add try/except statements to the command line interface so that the program can handle invalid entries.

**H. Verify that the data structure used in the solution meets all requirements in the scenario.**

The data structure used in this program–the hash table–meets all the requirements in the scenario. The hash table allows the efficient retrieval, insertion, and deletion of data. It stores each package in an index in the table using the package ID as a key, once stored the user can access all the information of the package using the corresponding key. It can also dynamically change to fit the size of the data set.

**1. Identify two other data structures that could meet the same requirements in the scenario.**

Two other data structures that could be used are lists and queues.

**a. Describe how each data structure identified in H1 is different from the data structure used in the solution.**

Lists store data in an ordered sequence, the main difference to a hash table is that list indices only store 1 item, whereas hash tables can store multiple items in a single index via chaining.

This results in the list having a longer search time compared to the hash table. This is also due to lists accepting duplicate items; therefore it must search through the entire list to find all the entries. Lists are better used in cases in which the order of the items being stored matter.

Similar to lists, queues are an ordered collection of items. Items are added at one end and removed from the other end; consequently, this means that the first item added to the queue will be the first one to be removed. The main difference between queues and hash tables are that items in queues can only be processed in the order they are entered.