Eric Jacobs
Student ID: 010580832
C950-Task-2

D:

```
ericjacobs@Erics-MacBook-Pro ~ % python3 -u "/Users/ericjacobs/Desktop/Software I/Project/C950/Main.py"
Truck 1 returning to hub, total miles: 24.200000000000003
Truck 2 returning to hub, total miles: 20.1
Truck 3 returning to hub, total miles: 28.2

Options:
1. Check Package Status
2. Display All Package Statuses
3. Display Truck Information
4. Display Total Mileage
5. Exit
Enter your choice: 2
Enter the time (HH:MM AM/PM) to check package statuses: 09:00 AM
Package ID: 1, Status: Delivered at 08:00 AM
Package ID: 2, Status: Delivered at 08:41 AM
Package ID: 3, Status: Delivered at 08:00 AM
Package ID: 4, Status: Delivered at 08:22 AM
Package ID: 5, Status: En Route
Package ID: 6, Status: En Route
Package ID: 7, Status: En Route
Package ID: 8, Status: Delivered at 08:41 AM
Package ID: 9, Status: En Route
Package ID: 10, Status: Delivered at 08:00 AM
Package ID: 11, Status: Delivered at 08:59 AM
Package ID: 12, Status: En Route
Package ID: 13, Status: Delivered at 08:30 AM
Package ID: 14, Status: Delivered at 08:00 AM
Package ID: 15, Status: Delivered at 08:00 AM
Package ID: 16, Status: Delivered at 08:00 AM
Package ID: 17, Status: En Route
Package ID: 18, Status: Delivered at 08:30 AM
Package ID: 19, Status: Delivered at 08:17 AM
Package ID: 20, Status: Delivered at 08:22 AM
Package ID: 21, Status: En Route
Package ID: 22, Status: En Route
Package ID: 23, Status: En Route
Package ID: 24, Status: En Route
Package ID: 25, Status: Delivered at 08:00 AM
Package ID: 26, Status: En Route
Package ID: 27, Status: En Route
Package ID: 28, Status: En Route
Package ID: 29, Status: Delivered at 08:59 AM
Package ID: 30, Status: Delivered at 08:41 AM
Package ID: 31, Status: Delivered at 08:28 AM
Package ID: 32, Status: En Route
Package ID: 33, Status: En Route
Package ID: 34, Status: Delivered at 08:06 AM
Package ID: 35, Status: En Route
Package ID: 36, Status: Delivered at 08:17 AM
Package ID: 37, Status: Delivered at 08:06 AM
Package ID: 38, Status: Delivered at 08:00 AM
Package ID: 39, Status: Delivered at 08:30 AM
Package ID: 40, Status: Delivered at 08:22 AM

Options:
1. Check Package Status
2. Display All Package Statuses
3. Display Truck Information
4. Display Total Mileage
5. Exit
Enter your choice:
```

```
Package ID: 31, Status: Delivered at 08:28 AM
Package ID: 32, Status: En Route
Package ID: 33, Status: En Route
Package ID: 34, Status: Delivered at 08:06 AM
Package ID: 35, Status: En Route
Package ID: 36, Status: Delivered at 08:17 AM
Package ID: 37, Status: Delivered at 08:06 AM
Package ID: 38, Status: Delivered at 08:00 AM
Package ID: 39, Status: Delivered at 08:30 AM
Package ID: 40, Status: Delivered at 08:22 AM

Options:
1. Check Package Status
2. Display All Package Statuses
3. Display Truck Information
4. Display Total Mileage
5. Exit
Enter your choice: 2
Enter the time (HH:MM AM/PM) to check package statuses: 10:00 AM
Package ID: 1, Status: Delivered at 08:00 AM
Package ID: 2, Status: Delivered at 08:41 AM
Package ID: 3, Status: Delivered at 08:00 AM
Package ID: 4, Status: Delivered at 08:22 AM
Package ID: 5, Status: En Route
Package ID: 6, Status: En Route
Package ID: 7, Status: En Route
Package ID: 8, Status: Delivered at 08:41 AM
Package ID: 9, Status: En Route
Package ID: 10, Status: Delivered at 08:00 AM
Package ID: 11, Status: Delivered at 08:59 AM
Package ID: 12, Status: En Route
Package ID: 13, Status: Delivered at 08:30 AM
Package ID: 14, Status: Delivered at 08:00 AM
Package ID: 15, Status: Delivered at 08:00 AM
Package ID: 16, Status: Delivered at 08:00 AM
Package ID: 17, Status: En Route
Package ID: 18, Status: Delivered at 08:30 AM
Package ID: 19, Status: Delivered at 08:17 AM
Package ID: 20, Status: Delivered at 08:22 AM
Package ID: 21, Status: En Route
Package ID: 22, Status: En Route
Package ID: 23, Status: En Route
Package ID: 24, Status: En Route
Package ID: 25, Status: Delivered at 08:00 AM
Package ID: 26, Status: En Route
Package ID: 27, Status: En Route
Package ID: 28, Status: En Route
Package ID: 29, Status: Delivered at 08:59 AM
Package ID: 30, Status: Delivered at 08:41 AM
Package ID: 31, Status: Delivered at 08:28 AM
Package ID: 32, Status: En Route
Package ID: 33, Status: En Route
Package ID: 34, Status: Delivered at 08:06 AM
Package ID: 35, Status: En Route
Package ID: 36, Status: Delivered at 08:17 AM
Package ID: 37, Status: Delivered at 08:06 AM
Package ID: 38, Status: Delivered at 08:00 AM
Package ID: 39, Status: Delivered at 08:30 AM
Package ID: 40, Status: Delivered at 08:22 AM

Options:
1. Check Package Status
2. Display All Package Statuses
3. Display Truck Information
4. Display Total Mileage
5. Exit
Enter your choice:
```

```
Package ID: 31, Status: Delivered at 08:28 AM
Package ID: 32, Status: En Route
Package ID: 33, Status: En Route
Package ID: 34, Status: Delivered at 08:06 AM
Package ID: 35, Status: En Route
Package ID: 36, Status: Delivered at 08:17 AM
Package ID: 37, Status: Delivered at 08:06 AM
Package ID: 38, Status: Delivered at 08:00 AM
Package ID: 39, Status: Delivered at 08:30 AM
Package ID: 40, Status: Delivered at 08:22 AM

Options:
1. Check Package Status
2. Display All Package Statuses
3. Display Truck Information
4. Display Total Mileage
5. Exit
Enter your choice: 2
Enter the time (HH:MM AM/PM) to check package statuses: 01:00 PM
Package ID: 1, Status: Delivered at 08:00 AM
Package ID: 2, Status: Delivered at 08:41 AM
Package ID: 3, Status: Delivered at 08:00 AM
Package ID: 4, Status: Delivered at 08:22 AM
Package ID: 5, Status: Delivered at 10:30 AM
Package ID: 6, Status: Delivered at 10:54 AM
Package ID: 7, Status: Delivered at 11:16 AM
Package ID: 8, Status: Delivered at 08:41 AM
Package ID: 9, Status: Delivered at 10:30 AM
Package ID: 10, Status: Delivered at 08:00 AM
Package ID: 11, Status: Delivered at 08:59 AM
Package ID: 12, Status: Delivered at 10:54 AM
Package ID: 13, Status: Delivered at 08:30 AM
Package ID: 14, Status: Delivered at 08:00 AM
Package ID: 15, Status: Delivered at 08:00 AM
Package ID: 16, Status: Delivered at 08:00 AM
Package ID: 17, Status: Delivered at 10:58 AM
Package ID: 18, Status: Delivered at 08:30 AM
Package ID: 19, Status: Delivered at 08:17 AM
Package ID: 20, Status: Delivered at 08:22 AM
Package ID: 21, Status: Delivered at 10:54 AM
Package ID: 22, Status: Delivered at 10:30 AM
Package ID: 23, Status: Delivered at 10:40 AM
Package ID: 24, Status: Delivered at 10:40 AM
Package ID: 25, Status: Delivered at 08:00 AM
Package ID: 26, Status: Delivered at 10:34 AM
Package ID: 27, Status: Delivered at 11:40 AM
Package ID: 28, Status: Delivered at 10:58 AM
Package ID: 29, Status: Delivered at 08:59 AM
Package ID: 30, Status: Delivered at 08:41 AM
Package ID: 31, Status: Delivered at 08:28 AM
Package ID: 32, Status: Delivered at 10:30 AM
Package ID: 33, Status: Delivered at 11:11 AM
Package ID: 34, Status: Delivered at 08:06 AM
Package ID: 35, Status: Delivered at 11:40 AM
Package ID: 36, Status: Delivered at 08:17 AM
Package ID: 37, Status: Delivered at 08:06 AM
Package ID: 38, Status: Delivered at 08:00 AM
Package ID: 39, Status: Delivered at 08:30 AM
Package ID: 40, Status: Delivered at 08:22 AM

Options:
1. Check Package Status
2. Display All Package Statuses
3. Display Truck Information
4. Display Total Mileage
5. Exit
Enter your choice:
```

E.

```
ericjacobs@Erics-MacBook-Pro ~ % python3 -u "/Users/ericjacobs/Desktop/Software I/Project/C950/Main.py"
Truck 1 returning to hub, total miles: 24.200000000000003
Truck 2 returning to hub, total miles: 20.1
Truck 3 returning to hub, total miles: 28.2

Options:
1. Check Package Status
2. Display All Package Statuses
3. Display Truck Information
4. Display Total Mileage
5. Exit
Enter your choice: 4
Total mileage by all trucks: 72.5

Options:
1. Check Package Status
2. Display All Package Statuses
3. Display Truck Information
4. Display Total Mileage
5. Exit
Enter your choice: 5
Exiting program.
ericjacobs@Erics-MacBook-Pro ~ %
```

F1:

Simplicity and Speed:
The Nearest Neighbor algorithm's primary strength lies in its simplicity, making it easy to implement and understand. It requires no complex data structures or intricate heuristics, making the development and debugging processes more straightforward. This simplicity also translates to speed, as the algorithm can quickly make delivery decisions, allowing for rapid progression through the delivery route without the need for extensive pre-computation or planning.

Local Optimization and Real-Time Decision Making:
Another significant advantage of the Nearest Neighbor algorithm is its ability to make real-time, locally optimized decisions. By always selecting the closest next destination, it minimizes the travel time between stops, potentially reducing the overall delivery time for packages. This local optimization is particularly effective when deliveries are clustered in certain areas, as it allows the algorithm to minimize travel distance incrementally.

Low Overhead:
The Nearest Neighbor algorithm operates with low overhead, as it does not require storing or managing large sets of possible routes. Instead, it dynamically selects the next step based on current conditions. This is advantageous for systems with limited computational resources or when the delivery parameters can change rapidly, such as in real-time traffic conditions or when dealing with last-minute delivery schedule changes.

In the context of the WGUPS Routing Program, these strengths align with the operational requirements of quick dispatch and the ability to adapt to changes throughout the delivery day. The decision to use the Nearest Neighbor algorithm is also justified by the scale of the operation, where the relatively small number of destinations does not necessitate the overhead of more complex algorithms that may offer global optimization but at the cost of increased complexity and computation time.

F2:

Effective Route Optimization:
The Nearest Neighbor algorithm adeptly aligns with the WGUPS routing program's need for efficient route planning. By prioritizing the closest unvisited locations for each delivery, this algorithm streamlines the delivery process, potentially reducing the overall time taken between consecutive deliveries.

Compatibility with Non-Linear Data Structures:
This algorithm seamlessly integrates with non-linear data structures like hash tables used in the program. The hash table effectively manages package data, including tracking and updating delivery statuses. The Nearest Neighbor algorithm leverages this data to make informed decisions on routing, thereby enhancing its efficiency.

Adherence to Operational Constraints: The Nearest Neighbor algorithm operates within the key constraints of the delivery system. It respects the truck capacity limits, ensuring that each truck's package load remains within the permissible 16-package maximum. Moreover, it aids in adhering to delivery deadlines by selecting the most immediately accessible destinations, thus minimizing the risk of delays.

Simple and Efficient Implementation: The algorithm’s simplicity makes it a practical choice for the WGUPS routing program. Its straightforward implementation ensures that the system remains robust and less prone to errors, which is critical in a real-world logistics operation.

Scalability and Flexibility: Despite its simplicity, the Nearest Neighbor algorithm offers a degree of scalability and flexibility. It can adapt to varying numbers of packages and different route configurations, making it a versatile tool for daily delivery operations.

Local Optimization for Quick Decisions: The algorithm's approach to local optimization, focusing on the immediate best choice without overcomplicating the overall route, enables quick and efficient decision-making. This aspect is particularly beneficial in dynamic delivery environments where time efficiency is paramount.

Facilitates Real-Time Adjustments: Given the real-time nature of package delivery, the Nearest Neighbor algorithm is apt as it allows for on-the-go route adjustments. This adaptability is crucial for handling unforeseen changes in delivery schedules or routes.

In summary, the Nearest Neighbor algorithm's use in the WGUPS routing program satisfies the scenario's requirements. Its ability to efficiently plan routes, work in tandem with complex data structures, and operate within the given logistical constraints, all while maintaining simplicity and adaptability, makes it a well-suited choice for the task at hand.

F3:

Dijkstra’s Algorithm:

Dijkstra’s algorithm excels in mapping out the shortest path from one origin to various destinations, which is beneficial for optimizing the route for a single truck from the hub to multiple delivery points. Unlike the Nearest Neighbor algorithm, which selects the next closest destination without considering future paths, Dijkstra's algorithm takes a broader view of the network, potentially finding a shorter overall path. This could result in substantial mileage savings, especially in a complex urban delivery network. However, for the given program, I prioritize simplicity and real-time decision-making, which Nearest Neighbor provides, over the complete route optimization that Dijkstra’s algorithm offers.

Dynamic Programming:

Dynamic programming approach would analyze all possible delivery sequences and choose the one with the optimal outcome, considering every package's constraints and delivery windows. This method is substantially different from the Nearest Neighbor algorithm, which does not consider future deliveries when choosing the next destination. Dynamic programming could potentially optimize the entire delivery day, but at the cost of increased initial computation time. Given that the package delivery system requires quick adjustments and real-time tracking, the Nearest Neighbor algorithm's simplicity and speed are more aligned with operational needs than the comprehensive but slower dynamic programming approach.

G:

Enhanced User Interface for Monitoring and Updates:

A more sophisticated user interface would significantly improve the user experience. Implementing a dashboard with real-time tracking, alerts for delivery updates, and interactive maps would allow users and supervisors to monitor progress effectively and make informed decisions.

Integration of Machine Learning for Predictive Analytics:

Machine learning models could be developed to predict traffic patterns, estimate delivery times more accurately, and suggest route alterations in response to unforeseen circumstances, such as road closures or delivery delays.

Use of API Integration for Real-Time Data: Integrating APIs from traffic and weather services could provide real-time data that can dynamically adjust routes to avoid delays, thereby improving punctuality and customer satisfaction.

H:
Efficient Access to Package Data:
The hash table structure in the WGUPS Routing Program excels in providing fast access to package information. Each package is uniquely identified by its ID, serving as the key in the hash table. This structure enables constant-time complexity (O(1)) for critical operations like inserting, retrieving, and updating package data, which is pivotal in a dynamic delivery environment.

Flexibility in Data Management: The hash table's design allows for flexibility in handling package data. For instance, when the address for package #9 needed updating at 10:20 AM, the hash table efficiently facilitated this change. This adaptability is essential for real-world scenarios where package information can change due to various factors.

Handling Multiple Package Attributes: The hash table effectively stores and manages various attributes for each package, such as delivery address, deadline, and status. This comprehensive data handling ensures that all necessary details are readily available for each package, enabling informed decision-making during the delivery process.

Compatibility with Routing Algorithm: The integration of the hash table with the Nearest Neighbor algorithm demonstrates a harmonious relationship between data storage and routing logic. The hash table provides the algorithm with quick access to package locations and statuses, aiding in efficient route calculation and package prioritization.

Scalability and Maintenance: The hash table's structure supports scalability, allowing the system to handle a growing number of packages without significant performance degradation. Moreover, the simplicity and clarity of the hash table's implementation contribute to easier maintenance and potential future enhancements of the program.

Resilience to Dynamic Changes: The hash table's ability to quickly adapt to changes, such as updating delivery statuses from 'At the hub' to 'En route' or 'Delivered', showcases its suitability for dynamic operational environments like package delivery, where statuses are frequently updated throughout the day.

In conclusion, the hash table used in the WGUPS Routing Program fulfills all the requirements set out in the scenario. Its efficiency in data retrieval and modification, combined with the ability to handle complex data structures and integrate seamlessly with the routing algorithm, makes it an ideal choice for this application.

H1:
Graphs:
A graph data structure consists of nodes (vertices) and edges, which naturally represent networks such as city maps and delivery routes, making it an ideal choice for routing problems. In a graph, each delivery location can be a node, with edges representing the direct paths between them, along with the distances as weights. This allows for the use of graph-specific algorithms like Dijkstra's or A* for finding the shortest path, which could potentially be more efficient than the nearest neighbor approach when dealing with complex routing scenarios. Graphs also facilitate the representation of various constraints and conditions of the delivery scenario, such as time windows, package dependencies, and truck capacities. Unlike hash

tables, which are excellent for key-value pair storage and retrieval, graphs excel in modeling relational data and spatial structures, providing a more holistic view of the relationships between different delivery points.

While graphs offer a visual and relational representation of the delivery network, the complexity of graph algorithms and the overhead of maintaining such structures may not be necessary for smaller scale routing problems. In the program, I opted for hash tables over graphs because they provide faster direct access for package data retrieval, which is crucial for real-time updates on package status, and are simpler to implement for the specific use case that does not require visual mapping of routes.

Binary Search Trees (BSTs):
A binary search tree is a hierarchical data structure where each node has at most two children, referred to as the left child and the right child. BSTs are sorted in a way that allows for efficient searching, insertion, and deletion operations that, on average, have a time complexity of O(log n). However, in the worst case, such as when the tree becomes unbalanced, these operations can degrade to O(n). In the context of package delivery, a BST could be used to organize packages by delivery deadlines or other sortable criteria, facilitating an ordered traversal that could assist in prioritizing deliveries. However, unlike hash tables which provide rapid access to package data via direct hashing, BSTs would require traversal from the root to the appropriate node, which can be less efficient for lookups. Balancing a BST to ensure its efficiency, such as in AVL trees or Red-Black trees, adds additional complexity to the implementation, which is not required for hash tables.

Although BSTs can maintain an ordered structure of packages, their performance can become inefficient in skewed or unbalanced scenarios, requiring rebalancing algorithms to maintain optimal search times. I chose hash tables in the program because they offer consistent average-time complexity for insertions, deletions, and lookups, which is more suitable for the dynamic nature of package tracking and doesn't necessitate the ordered structure that BSTs provide.