Kevin Colagiovanni
Student ID: 011039990

C950-DSA2 Task 1

A. A greedy algorithm has been chosen to create the program that delivers packages and is appropriate to use in this situation because it efficiently finds a short path between packages and it is also easy to implement and maintain. Because it is a greedy algorithm, it is not optimal, but it meets the requirements.

B. A hash table (a.k.a. dictionary) has been chosen to be used with the greedy algorithm to store the package data and is appropriate to use because it is self-adjusting, and has fast add and lookup times.

1. The hash table accounts for the relationship between the data components being stored using a list for each package. The package ID is used as the key, and the remaining package data is added to a list and that list is used as the value to the corresponding package ID (key).

C.

1. Greedy Algorithm (find_shortest_distance() method) Pseudo Code:

   Prerequisites:
   - The packages are already manually loaded onto the trucks based on their individual requirements (manual_load() method).

   Overview:
   - Start with the first truck
   - Iterate through all packages on the truck using a for loop and find the one that has the shortest distance to the hub and append the package_id to an empty list called “ordered_truck_list”, append the distance to an empty list called “total_truck_dist”, and calculate the travel time, based on the distance traveled, to an empty list called “delivery_times_list”.
   - Iterate through the package list using a for loop and find the package that has the shortest distance from it to the previous package that was appended to the “ordered_truck_list” and append it to the “ordered_truck_list”, append the distance to the “total_truck_dist” list, and calculate the travel time, based on the distance traveled, and append it to the “delivery_times_list” list. Continue doing that, each time finding the shortest distance from the current package to the last package that was last loaded until the end of the package list is reached.
   - Finally, get the distance from the last package to the hub and, append it to the “total_truck_dist” list, and calculate the travel time, based on the distance traveled, and append it to the “delivery_times_list” list.
   - Repeat steps 1.1 through 1.4 for subsequent trucks.

Pseudo Code (Python):

```
ordered_truck_list = []
truck_distance_list = []
truck_travel_time_list = []
loaded_package_list = []

for package in manually_loaded_truck:
    if ordered_truck_list is empty:
        minimum_distance = infinity
        for package in manually_loaded_truck_packages:
            if distance_to_hub < minimum_distance and package not in
        loaded_package_list:
                minimum_distance = distance_to_hub
                package_id = package
        ordered_truck_list.append(package_id)
        truck_distance_list.append(minimum_distance)
        truck_travel_time_list.append(calculate_travel_time(minimum_distance))
        loaded_package_list.append(package_id)

    minimum_distance = infinity
    for package in manually_loaded_truck_packages:
        if distance_to_previous_package < minimum_distance and package not
    in loaded_package_list:
        minimum_distance = distance_to_hub
        package_id = package
    ordered_truck_list.append(package_id)
    truck_distance_list.append(minimum_distance)
    truck_travel_time_list.append(calculate_travel_time(minimum_distance))
    loaded_package_list.append(package_id)

truck_distance_list.append(distance_from_last_package_to_hub)
truck_travel_time_list.append(calculate_travel_time(distance_from_last_package_to_hu
b))

check_package_requirements(order_truck_list) is True (requirements met):
```

2. PyCharm 2023.2.4 Community Edition was used for the IDE on a Lenovo ThinkPad P1 laptop with a Intel® Core™ i7-8850H CPU @ 2.60GHz × 12, 32GiB of RAM, 512GB SSD, builtin Intel graphics running two OS's (dual boot). Ubuntu 22.04.3 is primarily used, the secondary OS is Windows 10.

3. Space-Time Complexity:

    - Manual Loading: O(n^2)
    - Greedy algorithm: O(n^2)
    - Hash Table Class: O(n) when initializing and O(1) for each method
    - Over All program: O(n^2)
4. Because the time complexity is O(n^2) it will scale and adapt exponentially with a growing number of packages.
5. The software design will be easy and efficient to maintain because it is well commented and because it is broken down into smaller sections using multiple files, classes, and methods for various different tasks.
6. Strengths and weaknesses of a hash table (Geeks for Geeks, 28 Mar, 2023, Applications, Advantages and Disadvantages of Hash Data Structure)
    - Strengths:
        - Fast look up time.
        - Efficient lookup and deletion.
        - Space Efficiency.
        - Flexibility.
        - Collision Resolution.
    - Weaknesses:
        - Inefficient when there are a lot of collisions
        - Collisions are hard to avoid for a large set of possible keys.
        - A key cannot be a null value.
        - Limited capacity and will eventually fill up.
        - Can be complex to implement.
        - Orders of elements are not maintained, which makes it difficult to retrieve elements in specific order.

7. The Package ID was the choice of a key for efficient delivery management.

D. Reference List:

- Geeks for Geeks, Applications, Advantages and Disadvantages of Hash Data Structure, https://www.geeksforgeeks.org/applications-advantages-and-disadvantages-of-hash-data-structure/