Eric Jacobs
Student ID: 010580832
C950-Task 1

A:
I will be using the Nearest Neighbor Algorithm. This algorithm is well-suited for solving routing problems. It works by repeatedly choosing the closest next destination, which will build an efficient route step by step. It is simple and quick which make it well suited for this project.

B:
I will be using a hash table to store package data. This structure organizes data into key-value pairs. The package ID will be the key and the package details with be the value. This structure allows for rapid retrieval and updates of package information.

B1:
The hash table provides efficient lookup, inset and delete operations. The fact each package will have a Package ID # as a unique identifier will make good use of the hash table. The Package ID # will be used as at the key, while the package attributes will be used as the values in the Key-Value Pairs. The key (Package ID) will be the input for the hash function which output the index for the array.

C1:
```
Initialize packageHashTable
Initialize trucksList
Initialize distanceMatrix
Initialize addressIndexMap

Load addressIndexMap from 'Address.csv'
Load packageHashTable from 'Package.csv' using addressIndexMap
Load distanceMatrix from 'Distance.csv'

Assign packages to trucks based on nearest neighbor:
For each truck in trucksList:
   While truck has capacity and unassigned packages exist:
      nearestPackage = find nearest unassigned package to current location
      Assign nearestPackage to truck

Deliver packages using each truck:
For each truck in trucksList:
   Set current location to hub
   While truck has packages to deliver:
      nearestPackage = find nearest package to current location from truck's packages
      Deliver nearestPackage
      Update current location to nearestPackage's location
   Return truck to hub

Define function to find nearest unassigned package:
Function findNearestUnassignedPackage(currentLocation, packageHashTable, distanceMatrix):
   Initialize nearestPackage as null
   Initialize minDistance as infinity
   For each package in packageHashTable:
      If package is unassigned:
            distance = get distance from distanceMatrix using currentLocation and package's
location
            If distance < minDistance:
                Set nearestPackage to package
                Set minDistance to distance
    Return nearestPackage

Define function to find nearest package to deliver:
Function findNearestNeighbor(currentLocation, packagesList, distanceMatrix):
    Initialize nearestPackage as null
    Initialize minDistance as infinity
    For each package in packagesList:
        If package is not delivered:
            distance = get distance from distanceMatrix using currentLocation and package's
location
            If distance < minDistance:
                Set nearestPackage to package
                Set minDistance to distance
    Return nearestPackage and minDistance
```

C2:
I will be using Visual Studio Code on a macOS system. I am running python 3.12

C3:
Hash Table Operation: Insertion, lookup and delete would all have average case of O(1) and O(n) when collision occurs

Nearest Neighbor: Because the algorithm has to iterate over the list of packages for each delivery, due to it having to compare the distance between the current location and each of the remaining undelivered packages as a nested loop, it will be O(n^2).

Loading Packages onto Trucks: O(n) because each package needs to be viewed and potentially loaded.

Overall Program Complexity: O(n^2) because the Nearest Neighbor Search is the most time consuming part.

C4:
By using a hash table to manage package data, the program makes use of a scalable data structure. Because hash tables have an average-case time complexity of O(1) for insertion, lookup, and deletion operations, they can handle a large number of entries without bogging down the program.

The modular design of this program also for easy modification if wanting to scale up the production. Maybe we might want to add drone delivery to the program in the future, in which case, a drone class would be created along with with relevant CSV files and methods. To a lesser degree, the number of trucks, packages or addresses could be changed without having to overhaul the program. The CSV files work with this modular design.

The nearest neighbor algorithm will work well with changes in number of packages (scalability) or changes in delivery addresses (adaptability), such as with package #9. Because the program uses CSV files, it is able to be adapted to different cities, rather than being limited to just the Salt Lake City area. All that would have to be modified would be the connecting the program to new area's CSV files.

The algorithm doesn't require pre-calculated routes and is easy to setup so dynamically adjusting the number of packages doesn't effect it very much. It only needs to compare distances to the current locations, so additional packages don't add a lot of computation, even with time complexity of O(n^2).

C5:
This software design will be efficient due to a few factors. It will be designed modularly, which will allow updates and modifications of code without impacting the entire system. I will be implementing abstraction and encapsulation by creating separate classes for Truck, Package, and HashTable, which will further the modular design of the program.

The use of a hash table is ideal for package management because it allows for quick insertion, retrieval, and update operations. The hash table also allows for a growing number of packages, should this program be implemented on a larger or smaller scale.

The Nearest Neighbor Algorithm reduces the complexity of route planning.

I will also make sure the program is well commented and uses consistent naming conventions to enhance readability.

C6:
The strengths of the hash table are that it is offers fast access to package data and is efficient at handling a large number of entries.

The weaknesses of the hash table are that it has the potential for collisions, and memory usage can become high with a large number of entries. In this program we will attempt to mitigate the collisions using chaining.

C7:
The package ID is the most efficient key for delivery management. It uniquely identifies each package, allowing for quick access and updates to delivery status and details.