B. My Hashtable class has a lookup method where you put the package ID and returns all the package information. EX line 476 -🡪 print(hTable.lookUp(5))  
  
A screen shot of a computer

Description automatically generated

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*. [Package ID, Delivery Status, Truck#]*

**TIME: 09:03:00 AM  
  
A computer screen shot of a program

Description automatically generated**

1. 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. *[Package ID, Delivery Status, Truck#]*

**TIME: 10:06:06 AM**

**A screen shot of a computer

Description automatically generated**

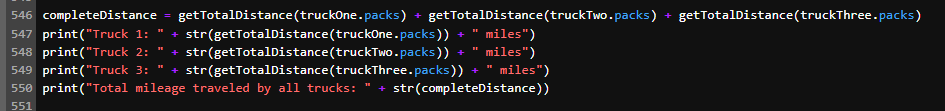
1. Provide screenshots to show the status of all packages loaded onto each truck at a time between 12:03 p.m. and 1:12 p.m.  *[Package ID, Delivery Status, Truck#]*

**TIME: 12:30:06 PM**

**A screen shot of a computer

Description automatically generated**

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



A screenshot of a computer

Description automatically generated

F1. Describe two or more strengths of the algorithm used in the solution:

The strengths about the algorithm that I chose – nearest neighbor algorithm. Would be that this algorithm is:

1. Responds quickly to change: by adding or removing packages, the code will still work as intended.
2. Fast and effective: no matter how large the data is, this algorithm will still be suitable.

F3. Identify two other named algorithms that are different from the algorithm implemented in the solution and would meet all requirements in the scenario:

* **heuristic algorithm** – we will start at the hub and find 16 packages that are closest to the hub. Those 16 packages will be in truck one. Then for truck 2, we will load another 16 packages that are closest to the hub that are not in truck 1. The rest will go to truck 3.
* **Dijkstra's algorithm** – I would have 3 paths for all 3 trucks. We would start with the 3 locations closest to the hub and end with 3 locations furthest to the hub. With this algorithm I would find the shortest paths from 1 of the closest location to one of the furthest locations.

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:

What I would do different is for the especial notes to be created with a specific number for each requirement, so then I can implement that requirement number into my coding.

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

- **List** – instead of a hash table, I would use the List data structure to add my packages  
This would be different because I would put the package ID in order and will make the same number as the index. With the hast table, the packages do not need to be in order.

- **Linked list** – where I would link the packages in each truck and the head would be the Hub

This would be different because with a linked list you can add and delete packages faster. Implementing packages into a linked list would also be faster.