A.  Identify a named self-adjusting algorithm (e.g., “Nearest Neighbor algorithm,” “Greedy algorithm”) that you used to create your program to deliver the packages.

B.  Write an overview of your program, in which you do the following:

1.  Explain the algorithm’s logic using pseudocode.

Note: You may refer to the attached “Sample Core Algorithm Overview” to complete part B1.

2.  Describe the programming environment you used to create the Python application.

3.  Evaluate the space-time complexity of each major segment of the program, and the entire program, using big-O notation.

4.  Explain the capability of your solution to scale and adapt to a growing number of packages.

5.  Discuss why the software is efficient and easy to maintain.

6.  Discuss the strengths and weaknesses of the self-adjusting data structures (e.g., the hash table).

C.  Write an original program to deliver all the packages, meeting all requirements, using the attached supporting documents “Salt Lake City Downtown Map,” “WGUPS Distance Table,” and the “WGUPS Package File.”

1.  Create an identifying comment within the first line of a file named “main.py” that includes your first name, last name, and student ID.

2.  Include comments in your code to explain the process and the flow of the program.

D.  Identify a self-adjusting data structure, such as a hash table, that can be used with the algorithm identified in part A to store the package data.

1.  Explain how your data structure accounts for the relationship between the data points you are storing.

Note: Use only appropriate built-in data structures, except dictionaries. You must design, write, implement, and debug all code that you turn in for this assessment. Code downloaded from the Internet or acquired from another student or any other source may not be submitted and will result in automatic failure of this assessment.

E.  Develop a hash table, without using any additional libraries or classes, that has an insertion function that takes the following components as input and inserts the components into the hash table:

•   package ID number

•   delivery address

•   delivery deadline

•   delivery city

•   delivery zip code

•   package weight

•   delivery status (e.g., delivered, en route)

F.  Develop a look-up function that takes the following components as input and returns the corresponding data elements:

•   package ID number

•   delivery address

•   delivery deadline

•   delivery city

•   delivery zip code

•   package weight

•   delivery status (i.e., “at the hub,” “en route,” or “delivered”), including the delivery time

G.  Provide an interface for the user to view the status and info (as listed in part F) of any package at any time, and the total mileage traveled by all trucks. (The delivery status should report the package as at the hub, en route, or delivered. Delivery status must include the time.)

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

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

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

H.  Provide a screenshot or screenshots showing successful completion of the code, free from runtime errors or warnings, that includes the total mileage traveled by all trucks.

I.  Justify the core algorithm you identified in part A and used in the solution by doing the following:

1.  Describe at least **two** strengths of the algorithm used in the solution.

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

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

a.  Describe how each algorithm identified in part I3 is different from the algorithm used in the solution.

J.  Describe what you would do differently, other than the two algorithms identified in I3, if you did this project again.

K.  Justify the data structure you identified in part D by doing the following:

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

a.  Explain how the time needed to complete the look-up function is affected by changes in the number of packages to be delivered.

b.  Explain how the data structure space usage is affected by changes in the number of packages to be delivered.

c.  Describe how changes to the number of trucks or the number of cities would affect the look-up time and the space usage of the data structure.

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

a.  Describe how each data structure identified in part K2 is different from the data structure used in the solution.