* [MY WGU](http://my.wgu.edu/" \o "My WGU Home)
* Uri Williams Easter

Ask for Help Print

NHP2 — NHP2 TASK 1: WGUPS ROUTING PROGRAM

**DATA STRUCTURES AND ALGORITHMS II — C950**

**PRFA — NHP2**

TASK OVERVIEWSUBMISSIONSEVALUATION REPORT

COMPETENCIES

**4048.5.1** : **Non-Linear Data**

The graduate creates software applications that incorporate non-linear data structures for efficient and maintainable software.

**4048.5.2** : **Hashing Algorithms and Structures**

The graduate writes code using hashing techniques within an application to perform searching operations.

**4048.5.3** : **Dictionaries and Sets**

The graduate incorporates dictionaries and sets in order to organize data into key-value pairs.

**4048.5.4** : **Self-Adjusting Data Structures**

The graduate evaluates the space and time complexity of self-adjusting data structures using big-O notation to improve the performance of applications.

**4048.5.5** : **Self-Adjusting Heuristics**

The graduate writes code using self-adjusting heuristics to improve the performance of applications.

**4048.5.6** : **NP-Completeness and Turing Machines**

The graduate evaluates computational complexity theories in order to apply models to specific scenarios.

INTRODUCTION

For this assessment, you will apply the algorithms and data structures you studied in this course to solve a real programming problem. You will also implement an algorithm to route delivery trucks that will allow you to meet all delivery constraints while traveling under 140 miles. You will then describe and justify the decisions you made while creating this program.

The skills you showcase in your completed project may be useful in responding to technical interview questions for future employment. This project may also be added to your portfolio to show to future employers.

SCENARIO

The Western Governors University Parcel Service (WGUPS) needs to determine an efficient route and delivery distribution for their Daily Local Deliveries (DLD) because packages are not currently being consistently delivered by their promised deadline. The Salt Lake City DLD route has three trucks, two drivers, and an average of 40 packages to deliver each day. Each package has specific criteria and delivery requirements.

Your task is to determine an algorithm, write code, and present a solution where all 40 packages (listed in the attached “WGUPS Package File”) will be delivered on time while meeting each package’s requirements and keeping the combined total distance traveled under 140 miles for both trucks. The specific delivery locations are shown on the attached “Salt Lake City Downtown Map,” and distances to each location are given in the attached “WGUPS Distance Table.” The intent is to use the program for this specific location and also for many other cities in each state where WGU has a presence*.* As such, you will need to include detailed comments to make your code easy to follow and to justify the decisions you made while writing your scripts.

Keep in mind that the supervisor should be able to see, at assigned points, the progress of each truck and its packages by any of the variables listed in the “WGUPS Package File,” including what has been delivered and at what time the delivery occurred.

ASSUMPTIONS

•   Each truck can carry a maximum of 16 packages, and the ID number of each package is unique.

•   The trucks travel at an average speed of 18 miles per hour and have an infinite amount of gas with no need to stop.

•   There are no collisions.

•   Three trucks and two drivers are available for deliveries. Each driver stays with the same truck as long as that truck is in service.

•   Drivers leave the hub no earlier than 8:00 a.m., with the truck loaded, and can return to the hub for packages if needed.

•   The delivery and loading times are instantaneous, i.e., no time passes while at a delivery or when moving packages to a truck at the hub (that time is factored into the calculation of the average speed of the trucks).

•   There is up to one special note associated with a package.

•   The delivery address for package #9, *Third District Juvenile Court*, is wrong and will be corrected at 10:20 a.m. WGUPS is aware that the address is incorrect and will be updated at 10:20 a.m. However, WGUPS does not know the correct address (410 S State St., Salt Lake City, UT 84111) until 10:20 a.m.

•   The distances provided in the WGUPS Distance Table are equal regardless of the direction traveled.

•   The day ends when all 40 packages have been delivered.

REQUIREMENTS

*Your submission must be your original work. No more than a combined total of 30% of the submission and no more than a 10% match to any one individual source can be directly quoted or closely paraphrased from sources, even if cited correctly. The similarity report that is provided when you submit your task can be used as a guide.*  
  
*You must use the rubric to direct the creation of your submission because it provides detailed criteria that will be used to evaluate your work. Each requirement below may be evaluated by more than one rubric aspect. The rubric aspect titles may contain hyperlinks to relevant portions of the course.*

*Tasks may****not****be submitted as cloud links, such as links to Google Docs, Google Slides, OneDrive, etc., unless specified in the task requirements. All other submissions must be file types that are uploaded and submitted as attachments (e.g., .docx, .pdf, .ppt).*

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.  
What is the algorithm chosen?

The Nearest Neighbor Algorithm was elected in this program as the solution for this traveling salesman problem. Nearest Neighbor finds the minimum distance between a grouping of points and maps to the closest proximal location using the smallest distance identified. This algorithm was one of the best choices given this problem since it serves the purpose of mapping an efficient path in terms of distance. This suggests that it will find one of the most optimized paths for delivering all packages quickly, and it will most likely stay within the limit for mileage for the day.

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. –email*

Truck\_list = [list of package ids on the truck]

Truck\_current\_location = 0

Min\_distance = some big number

Min\_package = None

While truck\_list: //will keep going while truck\_list is not empty

For pid in truck\_list:

          Get package from hash table using pid

          Current\_package\_location = address\_dict[package.address]

          Distance = distance\_list[truck\_current\_location][current\_package\_location]

          If distance < min\_distance:

                    Reset min\_distance and min\_package to current distance and package

//now you know the package to be delivered that’s closest to the current truck location

Calculate how many minutes it takes the truck to move to the package’s location

Add those minutes to the running time

Timestamp the package delivery time

Move the truck

Pop off the id of the package that was just delivered from truck\_list

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

# Pycharm 2023.2.2,

# Python 3.10.7

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

https://www.geeksforgeeks.org/how-to-analyse-loops-for-complexity-analysis-of-algorithms/

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

scalabity pros

Add package function

Clear package list

More trucks

Increase list size

can try expanding the hashmap use to add package objects to hashmap, and delete from hashmap

scalability cons

more addresses means we would need to switch from adjacency matrix as that isn't maintainable after a while

we could keep writing to csv file, but in reality, for industry we may want to connect to a database like sql, sqllite or mongodb, pymongo depending on app design and preference

writing to csv file is difficult to maintain because of overwriting

speed of truck constraints  
https://www.up.com/customers/track-record/tr081319-truck-pros-cons.htm#:~:text=Trucks%20travel%20at%20an%20average,and%20cost%2Deffective%20shipping%20solution.

chance of collisions with hashmap  
<https://www.geeksforgeeks.org/hash-map-in-python/>

need an algorithm to predict which truck should load which packages. Nn predicts nearest locations to each other, but given the complexity of constraints in this problem, for massive amounts of packages, and existing package limits, a more advanced prediction tool may be needed to allot a number of packages to the correct trucks.  
Inconveniences  
I knew the inconveniences because they were project requisites, in real life people most likely will not know many of the inconveniences faced ahead of time, nor will packages be delivered instantaneously and therefore delays in predicted delivery times will be slower.

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

Commenting, naming conventions, modular programming by defining a class for specific function set ,sources cited in code to explain where I get certain ideas from

<https://realpython.com/python-pep8/#naming-styles>

6.  Discuss the strengths and weaknesses of the self-adjusting data structures (e.g., the hash table).  
strengths O(1) time complexity, reduced space complexity  
weaknesses difficult to get size of all objects for scaling without a list or for loop as opposed to a dictionary

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.

--object instantiation from existing files (like csv), classes, and libraries (time e.g.)

--major algorithms that use the objects to generate data important to the requirements

--major UI functions that enable the user to evaluate the output of the major algorithms in a variety of contexts.

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.  
What structure was chosen? Chaining Hash table  
What does it do, and why did you think it was a good choice for this problem?

Uses calculated index to store several values to same location  
strengths O(1) time complexity, reduced space complexity

1.  Explain how your data structure accounts for the relationship between the data points you are storing.  
  
 Similar to a dictionary where the index corresponds to a location extracted from the distance files and the address files.   
What is a hashmap?

### Hash table overview

A ***hash table*** is a data structure that stores unordered items by mapping (or hashing) each item to a location in an array (or vector). Ex: Given an array with indices 0..9 to store integers from 0..500, the modulo (remainder) operator can be used to map 25 to index 5 (25 % 10 = 5), and 149 to index 9 (149 % 10 = 9). A hash table's main advantage is that searching (or inserting / removing) an item may require only O(1), in contrast to O(N) for searching a list or to O(log N) for binary search.

In a hash table, an item's ***key*** is the value used to map to an index. For all items that might possibly be stored in the hash table, every key is ideally unique, so that the hash table's algorithms can search for a specific item by that key.

Each hash table array element is called a ***bucket***. A ***hash function*** computes a bucket index from the item's key

What is a chaining hashmap?

***Chaining*** handles hash table collisions by using a list for each bucket, where each list may store multiple items that map to the same bucket. The insert operation first uses the item's key to determine the bucket, and then inserts the item in that bucket's list. Searching also first determines the bucket, and then searches the bucket's list. Likewise for removes  
The hash map uses modular arithmetic to store a series of objects in a designated location. In this instance a chaining HashMap was used to store several

In the Hashmap packages % 10 are stored in the same location for easy access.  
TravelData file  
--load\_address\_data reads in all rows from address csv  
--distance data variable loads all distances into a list of lists  
--Truck.py  
--calc\_distance function calculates distances in form of adjacency matrix to return the nearest positional value from within the distance\_data list

--address\_index function reads all rows within addressCSV and returns the index from column 0

in the nearest neighbor algorithm we use the attribute of the package id from the Truck’s list of package id’s known as package\_load. This enables us search for a package object within the hashmap using package id, and add it to a list of package objects.

Then while the list of package objects is greater than zero, we use the calc\_distance function to compare the address indices of the truck and packages to the nearest possible address, then we assign that address as the next address.  
Hashmap here plays a pivotal role in allowing the program to go from taking the id of the package, to looking up the package object so the addresses can be compared using the address\_index and calc\_distance function.

*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

*Note: Your function should output all data elements for the package ID number.*

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.

Nearest Neighbor  
No training step (knn does a training step)  
<https://quantdare.com/10-reasons-for-loving-nearest-neighbors-algorithm/>

Point pattern analysis model to find closest points nearest each other in space.  
<https://epgp.inflibnet.ac.in/epgpdata/uploads/epgp_content/S000017GE/P001787/M031066/ET/1527502623NNA_text(Final(1.pdf>

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

•   Each truck can carry a maximum of 16 packages, and the ID number of each package is unique.

•   The trucks travel at an average speed of 18 miles per hour and have an infinite amount of gas with no need to stop.

•   There are no collisions.

•   Three trucks and two drivers are available for deliveries. Each driver stays with the same truck as long as that truck is in service.

•   Drivers leave the hub no earlier than 8:00 a.m., with the truck loaded, and can return to the hub for packages if needed.

•   The delivery and loading times are instantaneous, i.e., no time passes while at a delivery or when moving packages to a truck at the hub (that time is factored into the calculation of the average speed of the trucks).

•   There is up to one special note associated with a package.

•   The delivery address for package #9, *Third District Juvenile Court*, is wrong and will be corrected at 10:20 a.m. WGUPS is aware that the address is incorrect and will be updated at 10:20 a.m. However, WGUPS does not know the correct address (410 S State St., Salt Lake City, UT 84111) until 10:20 a.m.

•   The distances provided in the WGUPS Distance Table are equal regardless of the direction traveled.

•   The day ends when all 40 packages have been delivered.

Answer with function processes and screenshots

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.

Christofides Algorithm

https://www.youtube.com/watch?v=ayIsRZAGyi4, https://www.youtube.com/watch?v=GiDsjIBOVoA&t=748s

Genetic algorithm

J.  Describe what you would do differently, other than the two algorithms identified in I3, if you did this project again.  
  
  
If this project were to be an industry facing solution to scale for over 4000 or more, there are in fact a few changes I would make. The first, is that I would add functions for the user to interact with where they could insert packages into the hashmap, or delete as needed if possible. My observations with big companies such as Fedex or Amazon is not that they merely load 4000 entries of data at a time, but rather any person who needs their services can submit a shipment order from wherever they are. Therefore, with an add button, that could help scale this project beyond its current capacity, and likewise for the delete, so users could undo unintentional orders. Further, the greater the amount of orders, I would try to dispatch more trucks. I would also see if there’s and api I could pull from to continue adjacency matrix for more locations, or see if another geolocator api could remedy the goal of working with shortest distances.  
Given a chance to do this again, I would also do more research into how to implement a Tkinter gui with this project. I was most of the way there, but I got stuck on how to display more rows into one of my treeviews.  
Finally, for the intended scale, I would setup a database. Most likely sql lite for value binding in queries and search.

--test different algorithms such as genetic algorithm + knn

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.  
not affected timewise, but space-wise, collisions can occur if one is not careful -

b.  Explain how the data structure space usage is affected by changes in the number of packages to be delivered.  
more buckets per hash\_index

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.  
--number of truck to package ratio could speed up the lookup time  
--greater number of cities could slow down lookup time in comparison of address index for calc distance. Less cities with greater distances could increase the distances the trucks travel, so the efficiency of the lookup function is determined by distance as well as number of cities.

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

--dictionary

(dictionary is to list as hashmap is to linked list)  
(both come from map and implement key value stores, but hashmap uses index)

https://www.reddit.com/r/learnpython/comments/s8oz0f/hash\_maps\_vs\_dictionaries/  
https://stackoverflow.com/questions/2061222/what-is-the-true-difference-between-a-dictionary-and-a-hash-table  
--list

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

--chaining hashmap can store multiple objects per index, no known length f(x)  
--dictionary maps key to value pairs, so it can be a little more difficult extracting object properties from the value when loaded from csv  
--list can store list of attributes per index, but has a greater number of indices to traverse as opposed to values per index.

L.  Acknowledge sources, using in-text citations and references, for content that is quoted, paraphrased, or summarized.

M.  Demonstrate professional communication in the content and presentation of your submission.

A, D, I,K, J. L

F, B,H

**File Restrictions**

File name may contain only letters, numbers, spaces, and these symbols: ! - \_ . \* ' ( )  
File size limit: 200 MB  
File types allowed: doc, docx, rtf, xls, xlsx, ppt, pptx, odt, pdf, txt, qt, mov, mpg, avi, mp3, wav, mp4, wma, flv, asf, mpeg, wmv, m4v, svg, tif, tiff, jpeg, jpg, gif, png, zip, rar, tar, 7z

RUBRIC

**A:**[**ALGORITHM IDENTIFICATION**](https://lrps.wgu.edu/provision/230479458)

|  |  |  |
| --- | --- | --- |
| **NOT EVIDENT**  A named algorithm is not identified. | **APPROACHING COMPETENCE**  The identified named self-adjusting algorithm does not perform the task of delivering the packages or does not perform the task. | **COMPETENT**  The identified named self-adjusting algorithm used to create the program performs the task of delivering *all* packages. |

**B1:**[**LOGIC COMMENTS**](https://lrps.wgu.edu/provision/230481127)

|  |  |  |
| --- | --- | --- |
| **NOT EVIDENT**  Pseudocode is not provided. | **APPROACHING COMPETENCE**  The submission provides pseudocode but does not explain the algorithm’s logic, or it contains inaccuracies. | **COMPETENT**  The submission explains the algorithm’s logic using pseudocode, and the algorithm’s logic contains *no* inaccuracies. |

**B2:DEVELOPMENT ENVIRONMENT**

|  |  |  |
| --- | --- | --- |
| **NOT EVIDENT**  A description is not provided. | **APPROACHING COMPETENCE**  The submission does not accurately describe the programming environment used to create the Python application. Or it includes either the software or hardware used in creating the program, but not *both*. | **COMPETENT**  The submission accurately describes the programming environment used to create the Python application, including *both* the software and hardware used to create the program. |

**B3:**[**SPACE-TIME AND BIG-O**](https://lrps.wgu.edu/provision/230481722)

|  |  |  |
| --- | --- | --- |
| **NOT EVIDENT**  An evaluation of space-time complexity is not provided. | **APPROACHING COMPETENCE**  The evaluation shows the space-time complexity of either major segments of the program or the entire program, but not *both*, using big-O notation. Or the evaluation addresses space or time, but not *both*. Or the evaluation contains inaccuracies. | **COMPETENT**  The evaluation accurately shows the space-time complexity of *each* major segment of the program, and the entire program, using big-O notation. |

**B4:**[**SCALABILITY AND ADAPTABILITY**](https://lrps.wgu.edu/provision/230482172)

|  |  |  |
| --- | --- | --- |
| **NOT EVIDENT**  The capability of the solution to scale and adapt to a changing market is not explained. | **APPROACHING COMPETENCE**  The capability of the solution to scale and adapt is explained, but the explanation is illogical, or it does not include the algorithm’s ability to scale to the number of packages. Or the explanation contains inaccuracies. | **COMPETENT**  The capability of the solution to scale and adapt is logically and accurately explained, including the algorithm’s ability to scale to the number of packages. |

**B5:**[**SOFTWARE EFFICIENCY AND MAINTAINABILITY**](https://lrps.wgu.edu/provision/230482172)

|  |  |  |
| --- | --- | --- |
| **NOT EVIDENT**  Neither the efficiency nor the maintainability of the software is discussed. | **APPROACHING COMPETENCE**  The discussion addresses either the efficiency or the maintainability of the software, but not *both*. Or the discussion contains inaccuracies. | **COMPETENT**  The discussion accurately addresses why the software is efficient and easy to maintain. |

**B6:**[**SELF-ADJUSTING DATA STRUCTURES**](https://lrps.wgu.edu/provision/230481127)

|  |  |  |
| --- | --- | --- |
| **NOT EVIDENT**  Neither the strengths nor the weaknesses of the self-adjusting data structures are discussed. | **APPROACHING COMPETENCE**  The discussion addresses either the strengths or the weaknesses of the self-adjusting data structures, but not *both*. Or the discussion contains inaccuracies. | **COMPETENT**  The discussion accurately addresses *both* the strengths and weaknesses of the self-adjusting data structures and the hash table. |

**C:ORIGINAL CODE**

|  |  |  |
| --- | --- | --- |
| **NOT EVIDENT**  Code is not provided. Or the code is not functional. | **APPROACHING COMPETENCE**  The code is original but does not run without errors or warnings, or the code is not original. | **COMPETENT**  The code is original and runs without errors or warnings. |

**C1:**[**IDENTIFICATION INFORMATION**](https://lrps.wgu.edu/provision/230482892)

|  |  |  |
| --- | --- | --- |
| **NOT EVIDENT**  An identifying comment is not provided. | **APPROACHING COMPETENCE**  The identifying comment is missing from the first line of a file named “main.py,” the file is not named “main.py,” or the comment is missing the candidate’s first name, last name, or student ID, or 1 or more of these elements is incorrect. | **COMPETENT**  An identifying comment is located within the first line of a file named “main.py” that includes the candidate’s correct first name, last name, and student ID. |

**C2:**[**PROCESS AND FLOW COMMENTS**](https://lrps.wgu.edu/provision/230482892)

|  |  |  |
| --- | --- | --- |
| **NOT EVIDENT**  Comments to explain the process and the flow are not provided. | **APPROACHING COMPETENCE**  Comments are provided separately from the code, or the comments do not adequately or accurately explain the process and the flow of the program. | **COMPETENT**  Comments are provided within the code that adequately and accurately explain the process and the flow of the program. |

**D:**[**DATA STRUCTURE**](https://lrps.wgu.edu/provision/230483583)

|  |  |  |
| --- | --- | --- |
| **NOT EVIDENT**  The submission does not identify a data structure. | **APPROACHING COMPETENCE**  The submission identifies a data structure, but the data structure is not self-adjusting, or it cannot store the package data. Or it does not perform well with the algorithm in part A. | **COMPETENT**  The submission identifies a self-adjusting data structure that performs well with the algorithm in part A and can store the package data. |

**D1:**[**EXPLANATION OF DATA STRUCTURE**](https://lrps.wgu.edu/provision/230483583)

|  |  |  |
| --- | --- | --- |
| **NOT EVIDENT**  The submission does not explain the data structure. | **APPROACHING COMPETENCE**  The submission explains the data structure but does not explain the relationship between the data points to be stored. Or the explanation contains inaccuracies. | **COMPETENT**  The submission accurately explains the data structure and how that data structure accounts for the relationship between the data points to be stored. |

**E:**[**HASH TABLE**](https://lrps.wgu.edu/provision/230483583)

|  |  |  |
| --- | --- | --- |
| **NOT EVIDENT**  A hash table is not provided. | **APPROACHING COMPETENCE**  The hash table does not have an insertion function or has an insertion function that includes additional libraries or classes or that does not account for *all* of the given components. Or the table contains errors. | **COMPETENT**  The hash table has an insertion function, without using *any* additional libraries or classes, that is free from errors and includes, as input, *all* of the given components. |

**F:**[**LOOK-UP FUNCTION**](https://lrps.wgu.edu/provision/230483583)

|  |  |  |
| --- | --- | --- |
| **NOT EVIDENT**  A look-up function is not provided. | **APPROACHING COMPETENCE**  The look-up function does not include *all* of the given data elements, or it does not complete searches or return the listed data. Or the look-up function does not complete or completes with run-time errors. | **COMPETENT**  The look-up function includes *all* of the given data elements, and it completes searches and returns the listed data. The look-up function completes without run-time errors. |

**G:INTERFACE**

|  |  |  |
| --- | --- | --- |
| **NOT EVIDENT**  An interface is not provided. | **APPROACHING COMPETENCE**  The interface provides an intuitive means for either determining the total mileage traveled by *all* trucks or for accessing package delivery status as required, but not *both*. Or the status check does not include the time. | **COMPETENT**  The interface provides an intuitive means of *both* determining the total mileage traveled by *all* trucks and for accessing package delivery status as required and includes the time of the status check. |

**G1:FIRST STATUS CHECK**

|  |  |  |
| --- | --- | --- |
| **NOT EVIDENT**  A screenshot is not provided. | **APPROACHING COMPETENCE**  The screenshots show an incomplete list of the packages that are loaded on *each* truck, or they do not show the status of *each* package at a time between 8:35 a.m. and 9:25 a.m. | **COMPETENT**  The screenshots show a list of *all* packages that are loaded on *each* truck and the status of *each* package at a time between 8:35 a.m. and 9:25 a.m. |

**G2:SECOND STATUS CHECK**

|  |  |  |
| --- | --- | --- |
| **NOT EVIDENT**  A screenshot is not provided. | **APPROACHING COMPETENCE**  The screenshots show an incomplete list of the packages that are loaded on *each* truck, or they do not show the status of *each* package at a time between 9:35 a.m. and 10:25 a.m. | **COMPETENT**  The screenshots show a list of *all* packages that are loaded on *each* truck and the status of *each* package at a time between 9:35 a.m. and 10:25 a.m. |

**G3:THIRD STATUS CHECK**

|  |  |  |
| --- | --- | --- |
| **NOT EVIDENT**  A screenshot is not provided. | **APPROACHING COMPETENCE**  The screenshots show an incomplete list of the packages that are loaded on *each* truck, or they do not show the status of *each* package at a time between 12:03 p.m. and 1:12 p.m. | **COMPETENT**  The screenshots show a list of *all* packages that are loaded on *each* truck and the status of *each* package at a time between 12:03 p.m. and 1:12 p.m. |

**H:SCREENSHOTS OF CODE EXECUTION**

|  |  |  |
| --- | --- | --- |
| **NOT EVIDENT**  Screenshots are not provided. | **APPROACHING COMPETENCE**  The screenshots capture an incomplete execution of the code, or the screenshot does not capture the total delivery time. | **COMPETENT**  The screenshots capture a complete execution of the code and include the total delivery mileage. |

**I1:**[**STRENGTHS OF THE CHOSEN ALGORITHM**](https://lrps.wgu.edu/provision/230479458)

|  |  |  |
| --- | --- | --- |
| **NOT EVIDENT**  A description is not provided. | **APPROACHING COMPETENCE**  *At least* 2 strengths of the algorithm identified in part A are described, but 1 or *both* of the strengths are inaccurate or do not relate to the scenario. | **COMPETENT**  *At least* 2 strengths specific to the algorithm identified in part A are accurately described, and *both* strengths apply to the scenario. |

**I2:VERIFICATION OF ALGORITHM**

|  |  |  |
| --- | --- | --- |
| **NOT EVIDENT**  The submission does not verify the algorithm used in the solution. | **APPROACHING COMPETENCE**  The submission does not verify the algorithm used in the solution meets *all* requirements. The verification is missing the total miles added to *all* trucks, or the total combined delivery distance is more than 140 miles, or it does not state that *all* packages were delivered on time. | **COMPETENT**  The submission verifies the algorithm used in the solution meets *all* requirements, and the verification includes the total miles added to *all* trucks, and the total combined delivery distance is less than 140 miles, and it states that *all* packages were delivered on time. |

**I3:**[**OTHER POSSIBLE ALGORITHMS**](https://lrps.wgu.edu/provision/230479458)

|  |  |  |
| --- | --- | --- |
| **NOT EVIDENT**  The submission does not identify 2 other algorithms. | **APPROACHING COMPETENCE**  The submission identifies 2 algorithms different from the one used in the solution, but 1 or *both* algorithms do not meet the requirements in the scenario. | **COMPETENT**  The submission identifies 2 algorithms different from the one used in the solution, and *both* algorithms meet the requirements in the scenario. |

**I3A:**[**ALGORITHM DIFFERENCES**](https://lrps.wgu.edu/provision/230479458)

|  |  |  |
| --- | --- | --- |
| **NOT EVIDENT**  The submission does not describe the differences between algorithms. | **APPROACHING COMPETENCE**  The submission does not describe attributes for *each* algorithm identified in part I3, or the description does not compare these attributes to the attributes of the algorithm used in the solution. Or the description contains inaccuracies. | **COMPETENT**  The submission accurately describes attributes of *each* algorithm identified in part I3, and it accurately compares these attributes to the attributes of the algorithm used in the solution. |

**J:DIFFERENT APPROACH**

|  |  |  |
| --- | --- | --- |
| **NOT EVIDENT**  A description is not provided. | **APPROACHING COMPETENCE**  The description identifies *at least* 1 aspect that would be done differently if the project were attempted again, but it does not describe what modifications would be made. | **COMPETENT**  The description identifies *at least* 1 aspect that would be done differently if the project were attempted again, and it includes details of the modifications that would be made. |

**K1:VERIFICATION OF DATA STRUCTURE**

|  |  |  |
| --- | --- | --- |
| **NOT EVIDENT**  The submission does not verify the data structure used in the solution. | **APPROACHING COMPETENCE**  The submission does not verify the data structure used in the solution meets *all* requirements. The verification is missing the total miles added to *all* trucks, or the total combined delivery distance is more than 140 miles, or *all* packages are not delivered on time, or the hash table with look-up function is missing, or the reporting needed is inaccurate or inefficient. | **COMPETENT**  The submission verifies the data structure used in the solution meets *all* requirements, and the verification includes the total miles added to *all* trucks, the total combined delivery distance is less than 140 total miles, *all* packages are delivered on time, the hash table with look-up function is present, and the reporting needed is accurate and efficient. |

**K1A:**[**EFFICIENCY**](https://lrps.wgu.edu/provision/230481722)

|  |  |  |
| --- | --- | --- |
| **NOT EVIDENT**  An explanation is not provided. | **APPROACHING COMPETENCE**  The explanation addresses how changes in the number of packages affect the time needed to complete the look-up function, but the explanation contains inaccuracies. | **COMPETENT**  The explanation accurately addresses how changes in the number of packages directly affect the time needed to complete the look-up function. |

**K1B:**[**OVERHEAD**](https://lrps.wgu.edu/provision/230481722)

|  |  |  |
| --- | --- | --- |
| **NOT EVIDENT**  An explanation is not provided. | **APPROACHING COMPETENCE**  The explanation addresses how changes in the number of packages affect the data structure space usage, but the explanation contains inaccuracies. | **COMPETENT**  The explanation accurately addresses how changes in the number of packages directly affect the data structure space usage. |

**K1C:**[**IMPLICATIONS**](https://lrps.wgu.edu/provision/230481722)

|  |  |  |
| --- | --- | --- |
| **NOT EVIDENT**  A description is not provided. | **APPROACHING COMPETENCE**  The description addresses how changes to the number of trucks or the number of cities would affect look-up time or space usage, but not *both*. Or the description contains inaccuracies. | **COMPETENT**  The description accurately addresses how changes to the number of trucks or the number of cities would affect look-up time and space usage. |

**K2:**[**OTHER DATA STRUCTURES**](https://lrps.wgu.edu/provision/230481722)

|  |  |  |
| --- | --- | --- |
| **NOT EVIDENT**  The submission does not identify other data structures. | **APPROACHING COMPETENCE**  The submission identifies only 1 data structure other than the one used in part D, or the submission identifies 2 data structures, but 1 or *both* of the data structures do not meet the requirements in the scenario. | **COMPETENT**  The submission identifies 2 data structures other than the one used in part D that could meet the requirements in the scenario. |

**K2A:**[**DATA STRUCTURE DIFFERENCES**](https://lrps.wgu.edu/provision/230481722)

|  |  |  |
| --- | --- | --- |
| **NOT EVIDENT**  The submission does not describe the differences between data structures. | **APPROACHING COMPETENCE**  The submission does not describe attributes for *each* data structure identified in part K2, or the description fails to compare these attributes to the attributes of the data structure used in the solution. Or the description contains inaccuracies. | **COMPETENT**  The submission accurately describes attributes of *each* data structure identified in part K2, and it accurately compares these attributes to the attributes of the data structure used in the solution. |

**L:**[**SOURCES**](https://lrps.wgu.edu/provision/147882373)

|  |  |  |
| --- | --- | --- |
| **NOT EVIDENT**  The submission does not include both in-text citations and a reference list for sources that are quoted, paraphrased, or summarized. | **APPROACHING COMPETENCE**  The submission includes in-text citations for sources that are quoted, paraphrased, or summarized and a reference list; however, the citations or reference list is incomplete or inaccurate. | **COMPETENT**  The submission includes in-text citations for sources that are properly quoted, paraphrased, or summarized and a reference list that accurately identifies the author, date, title, and source location as available. |

**M:**[**PROFESSIONAL COMMUNICATION**](https://lrps.wgu.edu/provision/27641407)

|  |  |  |
| --- | --- | --- |
| **NOT EVIDENT**  Content is unstructured, is disjointed, or contains pervasive errors in mechanics, usage, or grammar. Vocabulary or tone is unprofessional or distracts from the topic. | **APPROACHING COMPETENCE**  Content is poorly organized, is difficult to follow, or contains errors in mechanics, usage, or grammar that cause confusion. Terminology is misused or ineffective. | **COMPETENT**  Content reflects attention to detail, is organized, and focuses on the main ideas as prescribed in the task or chosen by the candidate. Terminology is pertinent, is used correctly, and effectively conveys the intended meaning. Mechanics, usage, and grammar promote accurate interpretation and understanding. |

SUPPORTING DOCUMENTS

Sample Core Algorithm Overview.docx

SLC downtown map.docx

WGUPS Package File.xlsx

WGUPS Distance Table.xlsx

Resource Center badge