**Data Structures and Algorithms II – C950**

**Course Overview**

Data Structures and Algorithms II explores the analysis and implementation of high-performance data structures and supporting algorithms, including graphs, hashing, self-adjusting data structures, set representations, and dynamic programming. The course also introduces students to NP-complete problems. The course discusses how to use Python techniques to implement software solutions for problems of memory management and data compression. This course has two prerequisites: Data Structures and Algorithms I and Discrete Math II.

**Objective Assessment Task Overview**

**COMPETENCIES**

**4048.5.1:** : **Non-Linear Data Structures**

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 implement an algorithm to route delivery trucks that will allow you to meet all delivery deadlines while traveling the least number of miles. You will also 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 the best 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 the best algorithm, write code, and present a solution where all 40 packages, listed in the attached “WGUPS Package File,” will be delivered on time with the least number of miles added to the combined mileage total of all 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.”

While you work on this assessment, take into consideration the specific delivery time expected for each package and the possibility that the delivery requirements—including the expected delivery time—can be changed by management at any time and at any point along the chosen route. In addition, you should 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 what time the delivery occurred.

The intent is to use this solution (program) for this specific location and to use the same program in many cities in each state where WGU has a presence*.* As such, you will need to include detailed comments, following the industry-standard Python style guide, to make your code easy to read and to justify the decisions you made while writing your program.

***Assumptions:***

•  Each truck can carry a maximum of 16 packages.

•  Trucks travel at an average speed of 18 miles per hour.

•  Trucks have a “infinite amount of gas” with no need to stop.

•  Each driver stays with the same truck as long as that truck is in service.

•  Drivers leave the hub at 8:00 a.m., with the truck loaded, and can return to the hub for packages if needed. The day ends when all 40 packages have been delivered.

•  Delivery time is instantaneous, i.e., no time passes while at a delivery (that time is factored into the average speed of the trucks).

•  There is up to one special note for each package.

•  The wrong delivery address for package #9, *Third District Juvenile Court*, will be corrected at 10:20 a.m. The correct address is 410 S State St., Salt Lake City, UT 84111.

•  The package ID is unique; there are no collisions.

•  No further assumptions exist or are allowed.

**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. An originality report is provided when you submit your task that 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.*

***Section 1: Programming/Coding***

A.  Identify the algorithm that will be used to create a program to deliver the packages and meets *all*  requirements specified in the scenario.

B.  Write a core algorithm overview, using the sample given, in which you do the following:

1.  Comment using pseudocode to show the logic of the algorithm applied to this software solution.

2.  Apply programming models to the scenario.

3.  Evaluate space-time complexity using Big O notation throughout the coding and for the entire program.

4.  Discuss the ability of your solution to adapt to a changing market and to scalability.

5.  Discuss the efficiency and maintainability of the software.

6.  Discuss the self-adjusting data structures chosen and their strengths and weaknesses based on the scenario.

C.  Write an original code to solve and to meet the requirements of lowest mileage usage and having *all*  packages delivered on time.

1.  Create a comment within the first line of your code that includes your first name, last name, and student ID.

2.  Include comments at *each*  block of code to explain the process and flow of the coding.

D.  Identify a data structure that can be used with your chosen algorithm to store the package data.

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

*Note: Do NOT use any existing data structures. 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, with 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, in 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 (e.g., delivered, in route)

G.  Provide an interface for the insert and look-up functions to view the status of any package at any time. This function should return *all* information about *each* package, including delivery status.

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

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

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

H.  Run your code and provide screenshots to capture the complete execution of your code.

***Section 2: Annotations***

I.  Justify your choice of algorithm by doing the following:

1.  Describe *at least* **two** strengths of the algorithm you chose.

2.  Verify that the algorithm you chose meets *all*  the criteria and requirements given in the scenario.

3.  Identify **two** other algorithms that could be used and would have met the criteria and requirements given in the scenario.

a.  Describe how *each*  algorithm identified in part I3 is different from the algorithm you chose to use in the solution.

J.  Describe what you would do differently if you did this project again.

K.  Justify your choice of data structure by doing the following:

1.  Verify that the data structure you chose meets *all*  the criteria and requirements given in the scenario.

a.  Describe the efficiency of the data structure chosen.

b.  Explain the expected overhead when linking to the next data item.

c.  Describe the implications of when more package data is added to the system or other changes in scale occur.

2.  Identify **two** other data structures that can meet the same criteria and requirements given in the scenario.

a.  Describe how *each*  data structure identified in part K2 is different from the data structure you chose to use in the solution.

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.

**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