MIS501 – Assessment 1 Case Study

The integration of online and mobile platforms presents a transformative opportunity for Transports Business, offering a comprehensive digital management solution. This initiative aims to boost online sales, streamline order management processes, and automate data flow for shipments. B2B, a prominent transport company in Australia, recognizes the evolving business landscape and seeks to meet customer demands with a convenient and contactless mobile ordering system.

As a member of the development team at SAS, your role involves designing and implementing the mobile ordering program. Each team member is tasked with completing ten predefined independent assignments, which will later converge to create the integrated platform. Prior to the application release deadline, the team must deliver the following ten programming tasks:

1. **Customer Details:**

Create a program that allows customers to enter their details to create a new account. Each customer should be asked to input the following information:

a. Customer's name

b. Mobile number

c. Address

d. Email address

e. Date of Birth

The program should validate the entered details and store them securely (Using appropriate Data Structure). If the user registration is successful (based on age is more than 21 and mobile number is 10 digit), display a success message with all the entered details; otherwise, display an appropriate error message.

2. **Vehicle Capacity:**

Write a program that advises the logistics manager on how many goods the company's vehicles can accommodate based on their dimensions. The program should ask the manager to input the length, width, and height of the vehicle in meters. If each cubic meter can hold 100 kilograms of goods, the program should calculate and output the maximum weight capacity of the vehicle.

Note: If the calculated weight capacity exceeds 5000 kilograms, the program should display a message indicating that the maximum weight capacity is 5000 kilograms.

3. **Delivery Time Estimation:**

Write a program that estimates the delivery time for a given route. The program should ask the user to input the distance of the route in kilometres and the average speed of the delivery vehicle in kilometres per hour. Using this information, the program should calculate and display the estimated delivery time in hours.

Note: If the calculated estimated delivery time is more than 15 hours, your code should take into account the driver rest time. Your program should add 8 hours rest to the delivery time and print the total estimated delivery time accordingly.

4. **Calculate Total Cost:**

Create a program that calculates and prints the total cost of a delivery based on the weight of the goods and the delivery distance. The program should ask the user to input the weight of the goods in kilograms and the delivery distance in kilometres. The cost per kilometre is $0.10 per kilogram. The minimum delivery charges are $30 and the program should apply a 5% discount for deliveries over 100 kilometres. The program should then calculate and display the total cost of the delivery.

Note: The program must roundoff the weight to the nearest positive value before calculating the cost of delivery.

5. **Route Optimization:**

Write a program that optimizes the delivery routes for multiple destinations. The program should ask the user to input the Name of 3 destinations and the distances to each destination from the warehouse in kilometres. Using this information, the program should design the route such that the driver visits all destinations based on the distance in a descending order(i.e. starting with the longest delivery first) and return to the starting point. The program should then display the optimized route.

Note: If 2 destinations have same distance then it must prioritize the destination that has been entered first.

6. **Goods Classification:**

Create a program that classifies goods into different categories based on their weight. The program should ask the user to insert the weight of a goods item in pounds. Based on the weight, the program should categorize the goods as follows:

- Lightweight: Less than 10 kilograms
- Mediumweight: More than 10 to 50 kilograms
- Heavyweight: More than 50 kilograms to 120 Kilograms
- Must be divided in small sizes: More than 120 Kilograms

The program should then display the category of the goods.

7. **Delivery Tracking:**

Write a program that tracks the delivery status of a package. The program should ask the user to input the package's tracking number and display the current status of the package, such as "In transit," "Out for delivery," or "Delivered."

Note: Use list to store and access the data for some sample tracking (A00001D) numbers and their status "In transit," "Out for delivery," or "Delivered." .

Note:

The program must ask the example of tracking number acceptable for checking the status.

The Program must reply with an appropriate message if the tracking number is not in the records or invalid format.

8. **Delivery Statistics:**

Create a program that calculates and displays the statistics of the company's deliveries. The program should ask the user to input the destination of 3 deliveries and the delivery times for each delivery. Using this information, the program should calculate and display the average delivery time and the fastest and slowest delivery times with destinations.

9. **Delivery Schedule:**

Write a program that generates a delivery schedule for the company's drivers. The program should ask the user to input the name of 3 drivers and their license number (in a prescribed and acceptable format). The program should then ask to enter the number of delivery available and then display a assigned number of deliveries of each driver, ensuring that each driver has an equal number of deliveries except the last one who can have more than others if needed.

10. **Account Settings:**

Implement a program that allows users to manage their account settings. Provide options to update their personal information (e.g., name, email address, password) and communication preferences (e.g., subscription to newsletters, promotional emails).

Note: Assume the account information are name, email address and password. The program saved the customer details in three list. A list comprises the names, another the email addresses and the third has the passwords. If a user has his name at position x in the first list, his email address will be found at the same position in the second lit and his password will be also saved in the same position in the third list. Write a program that takes as input the user' email address and password to validate and then enable the user to update personal information (email address must be unique) and communication preferences.

Each programming task should utilize concepts such as lists, tuples, conditionals, and input/output operations in Python.
