



COMSATS UNIVERSITY ISLAMABAD
DEPARTMENT OF COMPUTER SCIENCE

Final Term Examination, Spring 2025

Class: BAI/BCT-3

Subject: CSC211- Data Structures/Data Structures and Algorithms

Total Time Allowed: 180 minutes

Name: _____

Hammad Zaheer
SP24-BCT-027

Date: 01-July-2025

Instructor: Atique Ahmed Zafar

Max Marks: 50

Registration Number: _____

NOTE: Do as directed don't go into extra detail. Only crisp and concise answers will earn you credit

- Students are required to display their ID cards.
- Attempt all questions. Return this question paper with your answer sheet.
- Use of Mobile Phones and Digital Diaries not allowed

Q # 01

(CLO-1: Employ linear data structures to solve computing problems.)

Marks = [2*5 = 10]

You are tasked with designing a **Tab Manager** for a web browser that handles multiple open tabs. Each tab contains the following information:

- A unique Tab ID (e.g., 101)
- Website title (e.g., "YouTube")
- URL (e.g., "www.youtube.com")

To efficiently support navigation (Next/Previous tab), tab insertion and deletion, and full display of tab order, select an appropriate **Abstract Data Type (ADT)** and justify your selection through code. Implement the following operations using C++:

1.1 Write a function `openTab()` to open a new tab and add it to the tab list.

1.2 Write a function `closeTabByID()` to close an existing tab given its **Tab ID**.

(Ensure the tab manager maintains correct behavior after closing any tab, whether it is the active tab or not)

1.3 Write a function `switchTab(int TabID)` that switches directly to the tab with the given **Tab ID**, making it the new **active tab**.

1.4 Write a function `displayTabsForward()` that shows all open tabs in **forward** direction.

1.5 Write a function `displayTabsReverse()` that shows all open tabs in **reverse** direction.

(CLO-1: Employ linear data structures to solve computing problems.)

[Marks = [2*5 = 10]

02

You are tasked with designing an **Order Dispatch Manager** for a food delivery service (like *Foodpanda*) that handles and processes multiple active customer orders in real time. As customers place orders, the system must intelligently queue and dispatch them to available delivery riders while considering operational constraints like preparation time, item type, and rider availability. Urgent orders should be prioritized to ensure timely delivery and customer satisfaction. Each order contains the following information:

- Unique Order ID (e.g., 1001)
- Restaurant Name (e.g., "Burger Lab")
- Customer Location (e.g., "G-10 Islamabad")
- List of Items (e.g., "Burger, Fries, Coke")
- Estimated Preparation Time (in minutes)
- Time Order Was Placed (timestamp or counter)

You must use a static data structure, such as arrays or struct arrays, to manage insertion, dispatching, cancellation, and display of orders. **Do not** use dynamic data structures such as linked lists, STL vectors, or heap.

Implement the following operations using C++:

- 2.1 `addOrder()` – Add a new order; insert at the front (shifting existing orders right) if it contains only quick items (like drinks), otherwise at the end (normal enqueue).
- 2.2 `dispatchOrder()` – Dispatch the front order if a rider is available (simulate randomly); if an order has waited longer than its prep time, increase its priority by reordering it closer to the front (simulate priority using array rearrangement).
- 2.3 `viewNextOrder()` – Display the next eligible order for dispatch without removing it.
- 2.4 `displayAllOrders()` – Show all pending orders with details: ID, restaurant, location, items, time since placed, and urgency.
- 2.5 `cancelOrder(orderID)` – Cancel an order by its ID; ensure array elements are shifted properly to maintain order consistency.

(CLO-2: Use non-linear data structures to solve computing problems.)

Q # 03

Marks = [4+2+4= 10]

You are working with a company that processes real-time sales data, where each transaction includes a `transactionID`, `productName`, `amount`, and `timestamp`. A subset of these transactions is stored in a **Binary Search Tree (BST)** using the `amount` as the key. The current BST is built using the following transaction amounts:

[4500, 2200, 5000, 1800, 3000, 4700, 5100, 2600, 1900]

Due to a **customer refund** and a **manual correction**, the transactions with amounts **3000** and **5000** must be deleted.

- 3.1 Write C++ code to perform these deletions from the BST.
- 3.2 After deletions, draw and convert the resulting tree into a balanced AVL Tree using necessary rotations. Draw the final AVL Tree structure.

The business now wants to analyze high-value transactions for strategic sales insights, so transactions must be prioritized in descending order of amount.

- 3.3 Write C++ code to build a **Max Heap** from the AVL Tree and sort the transaction amounts in descending order.

Note: Do not use STL sort or heap libraries. Implement the heap manually.

(CLO-2: Use non-linear data structures to solve computing problems.)

Q # 04

Marks = [3+3+4= 10]

A city is planning the development of a smart traffic management system. The traffic control unit wants to model the network of intersections (nodes) and roads (edges) to optimize travel, connectivity, and cost. Each intersection is identified by a unique alphabetic ID (A-F). Roads have a cost (in minutes) reflecting congestion and signal delays. All roads are two-way, with symmetrical travel times. The goal is efficient communication and optimal routing for emergency vehicles.

Road Network (Adjacency List Representation):

Source Intersection	Connected Destinations (Neighbor, Travel Time in Minutes)
A	(B, 2), (C, 5), (E, 7)
B	(A, 2), (C, 1), (D, 4), (F, 8)
C	(A, 5), (B, 1), (D, 2), (E, 3)
D	(B, 4), (C, 2), (E, 2), (F, 6)
E	(A, 7), (C, 3), (D, 2), (F, 1)
F	(B, 8), (D, 6), (E, 1)

In broadcast urgent alerts from Intersection A, the system prioritizes reaching intersections with the minimum number of direct road segments (edges). The 'Time (minutes)' for roads is irrelevant for determining this specific broadcast order.

4.1 Choose and apply the most suitable graph traversal technique to determine the order intersections would receive this broadcast signal from A.

- Show: Chosen algorithm, justification (why suitable for "fewest edges"), step-by-step process (queue/visited state), and final visitation order.

The city needs a new communication cable network connecting all intersections with the minimum total installation cost, using 'Time (minutes)' as cost.

4.2 Apply Kruskal's Algorithm to construct the Minimum Spanning Tree (MST).

- Show: Edges sorted by cost, step-by-step Kruskal's (edge considered, decision, component state), final MST edges, and total minimum installation cost.

For emergency response, find the fastest possible routes from Intersection A to every other intersection, considering 'Time (minutes)' as travel time.

4.3 Apply Dijkstra's Algorithm, starting from Intersection A, to determine the minimum time to reach every other intersection.

- Show: Step-by-step Dijkstra's (node extracted, d/p updates for neighbors per iteration), and the final minimum time (cost) from A to B, C, D, E, and F.

(CLO-3 Apply fundamental sorting, searching, and hashing techniques on different data structures.)

Q # 05

Marks = [5+5=10]

5.1 A logistics company uses airport codes to track cargo dispatches. You are given the following airport codes:

ALPHA, BETA, GAMMA, DELTA, OMEGA, SIGMA, THETA, ZETA, LAMBDA, PHI, XI, RHO.

Hash function is defined as

$$h(\text{code}) = (\text{sum of ASCII values of first and third letters}) \% 17$$

Note: ASCII of 'A' = 65. Use this to find other uppercase letters.

- Create a hash table and show how the above keys are inserted using
 - Linear Probing
 - Separate Chaining
- Analyze number of collisions in each method.

5.2 You are working with a company that processes real-time sales data involving 50,000 transactions per day. Each transaction record includes, transactionID, productName, amount, and timestamp

Write an efficient C++ program to sort all transactions by:

- Descending order of amount
- For transactions with the same amount, use ascending order of timestamp

Note: The system receives transactions that are mostly entered in chronological order, with minor fluctuations in amount due to discounts, refunds, or manual edits occurring later in the day.

Best of Luck