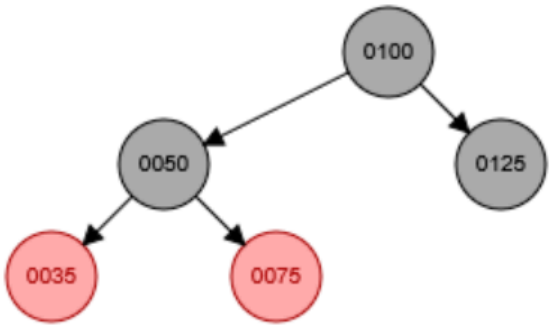
Quiz 2 Data Structure
Tuesday, 20 May 2025, 09.00 – 12.00 WIB (3 hours)
Odd Student Number
The quiz is a closed book. You only allow access to Ms. Word, Ms. Excel, Calculator and IDE.

A. Essay

1. [15 points] Given the following Red Black Tree:



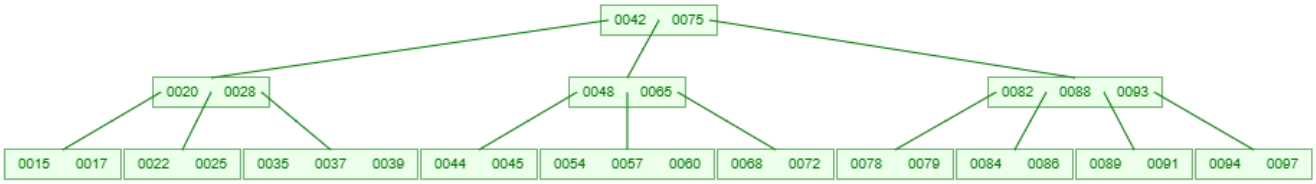
Using the existing Red Black Tree, simulate this process:

i. Insert 25
ii. Insert 10
iii. Insert 45
iv. Insert 40
v. Insert 37
vi. Delete 75
vii. Delete 45
viii. Delete 125
ix. Delete 25
x. Delete 100

Notes:

- After each operation, draw the resulting Red Black Tree
- Replace the delete value with maximum value from the left subtree when deleting the node with two children

2. [10 points] Given the following B-Tree:



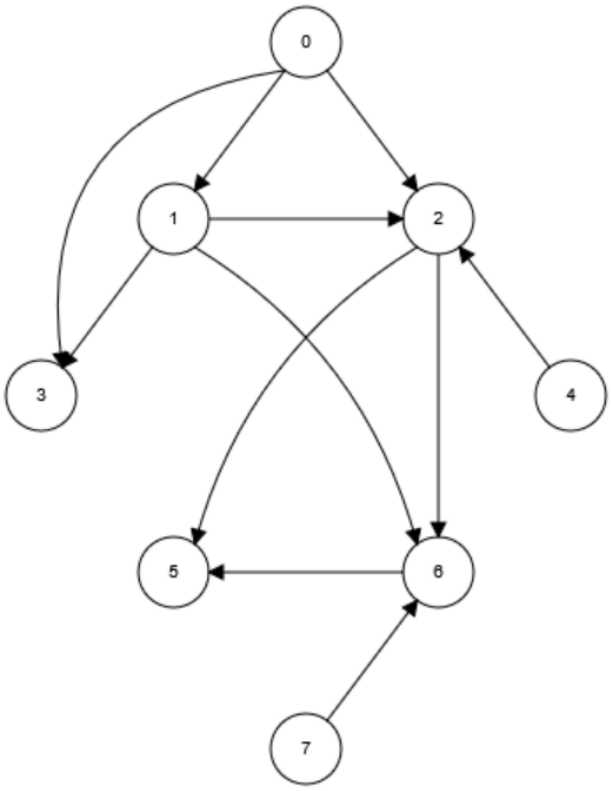
Using the existing **B-Tree order 5**, simulate this process:

i. Delete 94
ii. Delete 82
iii. Delete 20
iv. Insert 80
v. Insert 99

Notes:

- After each operation, draw the resulting B-Tree
- Replace the delete value with maximum value from the left subtree when deleting the node with two children

3. [5 points] Given the following Directed Graph:



Please perform path traversal using BFS and DFS **starting from node 0**

B. Case Study

1. [20 points] Heap

TechFlow Solutions manages thousands of daily computing tasks through their cloud infrastructure. These tasks range from critical security updates to routine maintenance. The system must process tasks based on urgency, where lower urgency scores indicate higher priority tasks.

The company needs a priority system that efficiently manages task execution. Each task has two properties:

- TaskName: Identifier for the task
- UrgencyScore: Priority value (lower = more urgent)

Input Format:

- The first line contains an integer **T** — the number of tasks.
- The next T lines each contain one operation:
  - *o* **ADD <TaskName> <UrgencyScore>**: Adds a new task to the system
    Example: ADD SecurityPatch 10 (adds a security patch with urgency score 10)
  - *o* **PROCESS**: Removes and returns the most urgent task (Highest priority = lowest urgency score)

Output Requirements:

- For each **PROCESS** operation:
  - Print the name of the task with lowest urgency score
  - Print "No jobs pending." if the task is empty

Constraints:

- 1 ≤ T ≤ 1000 (total operations)
- 1 ≤ Length of TaskName ≤ 30 (characters)
- 1 ≤ UrgencyScore ≤ 100 (lower score = higher priority)

Write a C program that implements this priority task management system using a **Min-Heap** structure.

Sample:

Input	Output
7 ADD BackupDB 30 ADD UpdateOS 15 ADD CleanCache 45 PROCESS ADD SecurityPatch 10 PROCESS PROCESS	UpdateOS SecurityPatch BackupDB

Explanation:

1. First 3 task added: BackupDB(30), UpdateOS(15), CleanCache(45)
2. PROCESS removes UpdateOS (lowest score: 15)
3. SecurityPatch(10) is added
4. PROCESS removes SecurityPatch (lowest score: 10)
5. PROCESS removes BackupDB (next lowest score: 30)

Input	Output
10 ADD SystemScan 25 PROCESS PROCESS ADD DataBackup 35 ADD SecurityUpdate 20 ADD NetworkCheck 40	SystemScan No jobs pending. SecurityUpdate DataBackup NetworkCheck No jobs pending.

PROCESS PROCESS PROCESS PROCESS	

Explanation:

1. SystemScan(25) is added and immediately processed
2. Second PROCESS finds empty
3. Three tasks added: DataBackup(35), SecurityUpdate(20), NetworkCheck(40)
4. Task processed in order of urgency: SecurityUpdate, DataBackup, NetworkCheck
5. Last PROCESS finds empty

2. [50 points] AVL Tree

FreshKeep Technologies has developed a smart refrigerator system called "SmartFridge Pro" that helps households and restaurants manage their food inventory efficiently. The system's primary goal is to minimize food waste by tracking expiration dates and maintaining optimal storage temperatures. The SmartFridge Pro needs an efficient system to:

- Monitor food items' expiration dates
- Alert users about soon-to-expire items
- Track inventory in real-time

Each item in the system must store:

1. **Item ID**: Unique identifier (e.g., 101, 102)
2. **Item Name**: Item name (e.g., Milk, Eggs)
3. **Expiration Date**: Format YYYY-MM-DD

FreshKeep Technologies decides to implement an automated system using **AVL Trees** to track and manage their perishable products effectively for the SmartFridge Pro

Input format:

- The first line contains an integer **N** – Number of items
- The next **N** lines each contain
  Item ID; Item Name; ExpiryDate separated by semi colons (ex: 102;Eggs;2025-05-15)
- The next line contains an integer **M** – Number of deletions
- The next **M** lines each contain
  Item ID to be deleted
- The last 2 lines contains:
  [Search Start Expired Date]
  [Search End Expired Date]

Output format:

- Display all items that fall within the given date range.

- For each item, display the date and items' name in ascending order (from the earliest to the latest expired) with following format:
  Expired Date – Item Name (ex: 2025-05-20 – Milk)

Constraint:
1 ≤N,M ≤1000
All dates are guaranteed to be valid (yyyy-MM-dd format).
The maximum length of an item's name is 30 characters.
All Item ID are guaranteed to be unique.
Search ranges are guaranteed to be valid (the start date is earlier than the end date)

Sample:

Input	Output
5 101;Milk;2025-05-20 102;Eggs;2025-05-15 103;Cheese;2025-05-25 104;Yogurt;2025-05-18 105;Butter;2025-05-22 2 102 104 2025-05-15 2025-05-20	2025-05-20 - Milk

Explanation:
1. Initial items: Milk, Eggs, Cheese, Yogurt, Butter
2. Delete items: Eggs(102), Yogurt(104)
3. Search range: 2025-05-15 to 2025-05-20
Only Milk shown (Eggs and Yogurt were deleted)

Input	Output
6 201;Apple;2025-06-01 202;Banana;2025-06-05 203;Orange;2025-06-10 204;Grape;2025-06-15 205;Mango;2025-06-12 206;Kiwi;2025-06-08 1 203 2025-06-05 2025-06-10	2025-06-05 - Banana 2025-06-08 - Kiwi

Explanation:
1. Initial items: Apple, Banana, Orange, Grape, Mango, Kiwi

2. Delete item: Orange(203)
3. Search range: 2025-06-05 to 2025-06-10
Show Banana and Kiwi (Orange was deleted)