# Lab 5

## Problem 1

You are given an array of integers `arr[]`, where the elements first increase, reach a peak, and then decrease. The peak element is defined as an element that is greater than or equal to its neighbors. The array guarantees that there is exactly one peak element. Your goal is to determine the index of the peak element in the array.

You should solve this problem using an efficient approach with time complexity **O(log n)**. Justify your answer based on this input `[1, 2, 4, 5, 7, 8, 3]`

---

## Problem 2

You are given two sorted arrays `a[]` and `b[]`. Your task is to find and return the **median** of the combined array formed by merging `a[]` and `b[]`.

The median is the middle value in an odd-length array, or the average of the two middle values in an even-length array.

You must solve this problem using an efficient approach with a time complexity of **O(log(min(n, m)))**, where `n` and `m` are the lengths of the two arrays.

Justify your answer based on these inputs `a[] = [1, 3, 8]` and `b[] = [7, 9, 10, 11]`)

---

## Problem 3

You are given a hash table with **5 buckets** (indexed from `0` to `4`).
Each bucket uses **separate chaining** to handle collisions (i.e., each bucket contains a linked list of records).

The **hash function** is defined as: h(key) = key % 5

Each record contains a **student ID** and **name**.

**Initial Insertions:**

The following student records are inserted **in order**:

| Student ID | Name |
|---|---|
| 7 | Alice |
| 12 | Bob |
| 17 | Carol |
| 3 | David |
| 8 | Eva |

**Tasks:**

1) **Build the Hash Table**
   a) Use the given hash function to place each student into the correct bucket.
   b) For each insertion, **show the bucket index** and whether a **collision occurs**.
   c) Draw the **final state** of the hash table, showing linked lists where collisions happen.

2) **Perform Operations** Using the final hash table:
   a) **Search for student ID 12**:
      - Which bucket is accessed?
      - How is the student found?

   b) **Insert a new student**: (ID = 22, Name = Frank)
      - Which bucket does this go to?
      - Does a collision occur?
      - Update the hash table accordingly.

   c) **Delete student ID 17**:
      - Which bucket is accessed?
      - Explain how the record is removed from the linked list.
      - Show the updated state of the bucket.

---

## Problem 4

You are tasked with selecting a **hash table collision resolution technique** for an application that stores **user IDs** as keys and **user information** as values. The two techniques you are considering are:

1. **Separate Chaining (Open Hashing)**: Uses linked lists for handling collisions.
2. **Open Addressing (Closed Hashing)**: Resolves collisions by searching for the next available slot.

**Task:**

1. **Efficiency Analysis**: Which technique is more efficient when the hash table is **sparsely populated** (low load factor) versus when it is **nearly full** (high load factor)? Discuss their performance in these two scenarios.
2. How does each technique perform in scenarios with **frequent deletions**? Consider how the deletion operation might affect **open addressing** (e.g., clustering issues).
3. If the application involves **frequent insertions**, which collision resolution technique would you recommend? Why?

---

## Problem 5

Imagine you're part of a company's performance evaluation team. The organization is structured hierarchically like a **binary tree**, where:

- Each **node** represents an **employee**, including team leads and individual contributors.
- Each **employee node** contains a **performance score** for a specific quarter.
- A **leaf node** is an individual contributor with no subordinates.
- **Internal nodes** (team leads or managers) manage up to two direct reports (left and right child in the tree).

Now, to **validate fair performance reporting**, the company defines a rule:

> "Each team lead's performance score must equal the **total performance score** of their **entire team** (i.e., the sum of their direct subordinates' performance scores and all levels below)."

Such a tree is called a **"Sum Tree"**.

**Task:**

Your job is to **verify** whether the company's hierarchy satisfies the **Sum Tree property**, i.e., whether the performance score of every team lead equals the **sum of the scores of all their subordinates**.

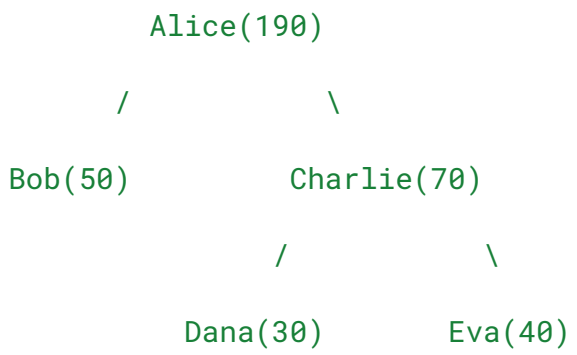
You must describe:

- The **algorithm runs in O(n) time complexity** you'd use to validate the Sum Tree property.
- A **step-by-step walkthrough** using the provided example tree.

**Note:**

- A leaf employee (individual contributor) is considered valid by default.
- An empty team (no employees) is considered to contribute 0 to the total.

**Example Hierarchy (Binary Tree):**




**Expected Output:** Yes, this is a valid Sum Tree.

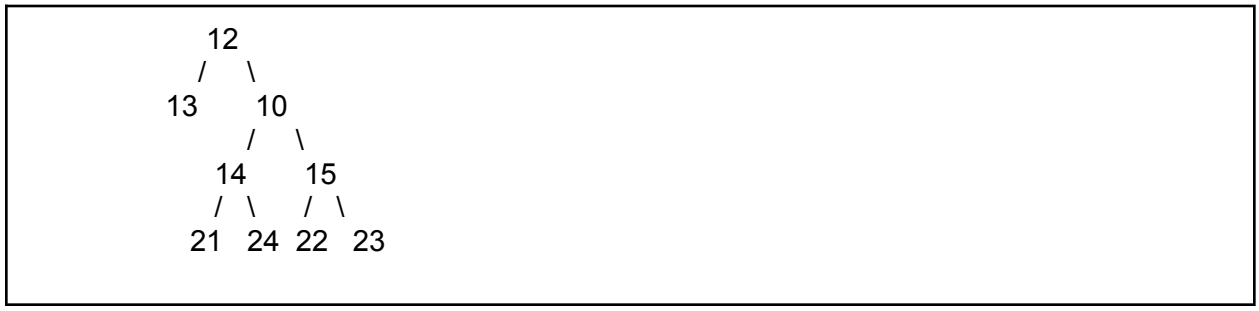
---

## Problem 6

We have a network of devices represented by a binary tree, where each device is connected to others, with parent-child relationships indicating direct connections. When **a failure** (such as a network outage, malfunction, or device failure) **occurs at a particular device** (the target node), **the network can no longer reach the failed device or its directly connected devices.** Our goal is to determine the order in which devices fail as the failure progresses throughout the network. By determining this order, we can understand how the failure will affect connectivity, identify critical devices whose failure could cause widespread disruption, and prioritize actions to prevent further failures or mitigate their impact on the overall network.

1. **Describe the Algorithm** you would use to simulate the failure spreading through the binary tree.
2. **Justify your approach** by showing the steps with the following example.

**Example Tree:**




Target Node = 14

**Expected Output:**

```
14

21, 24, 10

15, 12

22, 23, 13
```