**Name**: Guruphale Snehal Dileep (**230380720037**)

Case Study: Memory Management in Operating System and Working of Garbage Collection with Java using Custom Linked List

**Background:**

Memory management is a critical aspect of an operating system that ensures efficient utilization of computer memory. In Java, garbage collection is a mechanism that automates memory management, freeing developers from manual memory deallocation. In this case study, we will explore memory management in an operating system, understand garbage collection in Java, and implement a custom data structure using decomposition, pattern recognition, abstraction, and algorithm.

**Objective:**

The main objective of this case study is to demonstrate memory management concepts in an operating system and the working of garbage collection in Java. We will create a custom linked list data structure using a step-by-step approach that involves decomposition, pattern recognition, abstraction, and algorithm.

**Step 1: Decomposition**

Decomposition is the process of breaking down a complex problem into smaller, more manageable parts. In this step, we will identify the main components of our custom linked list:

- Data Structure: Custom Linked List

- Class: Node

- Class: LinkedList

- Methods: insert, delete, display

**Step 2: Pattern Recognition**

Pattern recognition involves identifying similarities in various parts of the problem. In this step, we will determine patterns for data storage and manipulation in our custom linked list data structure:

- Data Storage: We will use nodes to store elements in the linked list.

- Data Manipulation: We will use methods to insert and delete nodes and display the linked list.

**Step 3: Abstraction**

Abstraction involves representing objects or ideas at a higher, more generalized level, without worrying about the specific implementation. In this step, we will define the abstract structure of our custom linked list:

```java

class Node {

int data;

Node next;

Node(int data) {

this.data = data;

this.next = null;

}

}

class LinkedList {

private Node head;

// Methods for custom linked list:

// - insert

// - delete

// - display

}

```

**Step 4: Algorithm**

An algorithm is a step-by-step procedure to solve a specific problem. In this step, we will outline algorithms for the insert, delete, and display methods of our custom linked list:

- Algorithm for insert method:

- Input: Data to be inserted

- Create a new node with the given data

- Set the next pointer of the new node to the current head

- Set the head to the new node

- Algorithm for delete method:

- Input: Data to be deleted

- Search for the node with the given data in the linked list

- If found:

- Set the next pointer of the previous node to the next node of the node to be deleted

- If not found, do nothing

- Algorithm for display method:

- Start from the head node

- Traverse through each node in the linked list

- Display the data of each node

**Conclusion:**

Memory management in an operating system is essential for efficient resource usage. Java's garbage collection automates memory management, freeing developers from manual memory deallocation. In this case study, we implemented a custom linked list data structure using decomposition, pattern recognition, abstraction, and algorithm. Understanding memory management and garbage collection is crucial for creating efficient and reliable applications.