Day 12

Assignments 2: Linked List Middle Element Search
You are given a singly linked list. Write a function to find the middle element without using any extra space and only one traversal through the linked list.

A)
Introduction:

To find the middle element of a singly linked list in one traversal without using extra space, you can use the "two-pointer technique". This technique involves two pointers, slow and fast. Here's how it works:

Initialize both pointers to the head of the linked list.

Move the fast pointer two steps and the slow pointer one step at a time.

When the fast pointer reaches the end of the list, the slow pointer will be at the middle.

This method ensures that you only traverse the list once and use a constant amount of space.

Here's a Java implementation of this approach:

```
class LinkedList {
   Node head; // head of the list

   // Linked list node
   static class Node {
      int data;
      Node next;

      Node(int d) {
         data = d;
         next = null;
      }
   }

   // Function to print the middle of the linked list
   void printMiddle() {
      Node slow = head;
      Node fast = head;

      if (head != null) {
         while (fast != null && fast.next != null) {
            fast = fast.next.next;
            slow = slow.next;
         }
         System.out.println("The middle element is: " + slow.data);
```

```
        }
    }

    // Function to add a new node at the end of the list
    public void append(int new_data) {
        Node new_node = new Node(new_data);

        if (head == null) {
            head = new_node;
            return;
        }

        Node last = head;
        while (last.next != null) {
            last = last.next;
        }
        last.next = new_node;
    }

    // Function to print the linked list
    public void printList() {
        Node tnode = head;
        while (tnode != null) {
            System.out.print(tnode.data + " -> ");
            tnode = tnode.next;
        }
        System.out.println("null");
    }

    public static void main(String[] args) {
        LinkedList list = new LinkedList();
        for (int i = 1; i <= 9; i++) {
            list.append(i);
        }
        list.printList();
        list.printMiddle();
    }
}
```

Output:

When you run the provided code with the example linked list (containing elements 1 to 9), the output will be:

```
1 -> 2 -> 3 -> 4 -> 5 -> 6 -> 7 -> 8 -> 9 -> null

The middle element is: 5
```

Explanation:

Node class: Defines the structure of each node in the linked list.

LinkedList class: Contains methods to manipulate the linked list.

append(int new_data): Adds a new node at the end of the list.

printList(): Prints all elements of the list.

printMiddle(): Finds and prints the middle element using the two-pointer technique.

main method: Demonstrates the usage of the LinkedList class by creating a list, appending elements, printing the list, and finding the middle element

Summary:

This method is easy to understand and implement. It involves two traversals of the list and uses no additional space other than a few variables. While not as efficient as the two-pointer technique, it's straightforward and suitable for educational purposes or simple applications where ease of understanding and implementation is prioritized.