❖ Middle Element:

- Explanation:
  The program finds the middle element of a linked list using the two-pointer technique, where one pointer moves at twice the speed of the other. When the fast pointer reaches the end of the list, the slow pointer will be at the middle node. This approach efficiently identifies the middle element in a single pass through the list.
- Time Complexity and Space Complexity:
  - O(n)   O(1)
- Flowchart:

```
[Start]
   |
[Input Linked List]
   |
[Head is null?] -- Yes --> [Output: "The list is
empty."] --> [End]
   | No
   |
[Initialize Slow Pointer to Head]
   |
[Initialize Fast Pointer to Head]
   |
[Fast is not null and Fast.next is not null?] -- No -->
[Output Middle Element (Slow Pointer)] --> [End]
   | Yes
   |
[Move Slow to Slow.next]
   |
[Move Fast to Fast.next.next]
    |
[Back to Fast is not null and Fast.next is not null?]
```