Task 1: Code access, searching, insertion, and deletion in linked lists.

Task 2a: Given a singly linked list, reverse the list such that the head is the new tail and the tail is the new head. All nodes' previous will now be its next.

Example: head → 1 → 2 → 3 → 4 → null will become head → 4 → 3 → 2 → 1 → null

Task 2b: Given a doubly linked list, reverse the list.

Possible implementation: Recursive solution for Task 2b: Divide the list in two parts - first node and rest of the linked list. Call reverse for the rest of the linked list. Link rest to first. Fix head pointer.

Task 1: Code access, searching, insertion, and deletion in linked lists.

Task 2a: Given a singly linked list, reverse the list such that the head is the new tail and the tail is the new head. All nodes' previous will now be its next.

Example: head → 1 → 2 → 3 → 4 → null will become head → 4 → 3 → 2 → 1 → null

Task 2b: Given a doubly linked list, reverse the list.

Possible implementation: Recursive solution for Task 2b: Divide the list in two parts - first node and rest of the linked list. Call reverse for the rest of the linked list. Link rest to first. Fix head pointer.

Task 1: Code access, searching, insertion, and deletion in linked lists.

Task 2a: Given a singly linked list, reverse the list such that the head is the new tail and the tail is the new head. All nodes' previous will now be its next.

Example: head → 1 → 2 → 3 → 4 → null will become head → 4 → 3 → 2 → 1 → null

Task 2b: Given a doubly linked list, reverse the list.

Possible implementation: Recursive solution for Task 2b: Divide the list in two parts - first node and rest of the linked list. Call reverse for the rest of the linked list. Link rest to first. Fix head pointer.

Task 1: Code access, searching, insertion, and deletion in linked lists.

Task 2a: Given a singly linked list, reverse the list such that the head is the new tail and the tail is the new head. All nodes' previous will now be its next.

Example: head → 1 → 2 → 3 → 4 → null will become head → 4 → 3 → 2 → 1 → null

Task 2b: Given a doubly linked list, reverse the list.

Possible implementation: Recursive solution for Task 2b: Divide the list in two parts - first node and rest of the linked list. Call reverse for the rest of the linked list. Link rest to first. Fix head pointer.