```
class Node:
    """Node class to represent individual nodes in the linked list"""
    def __init__(self, data):
        self.data = data
        self.next = None

class LinkedList:
    """LinkedList class to manage the nodes and operations"""
    def __init__(self):
        self.head = None   # empty list

    def add_node(self, data):
        """Add a node to the end of the list"""
        new_node = Node(data)  # Create a new node


        if self.head is None:
            self.head = new_node
        else:

            current = self.head
            while current.next is not None:
                current = current.next
            # Add new node at the end
            current.next = new_node

    def print_list(self):
        """Print all elements in the linked list"""
        current = self.head
        elements = []

        # Traverse through the list and collect data
        while current is not None:
            elements.append(str(current.data))
            current = current.next

        # Print the list or indicate it's empty
        if elements:
            print(" -> ".join(elements))
        else:
            print("The list is empty")

    def delete_nth_node(self, n):
        """
        Delete the nth node (1-based index)

        Args:
            n (int): The position to delete (starting from 1)

        Raises:
            ValueError: If list is empty or index is invalid
        """
        # Check for empty list
        if self.head is None:
            raise ValueError("Cannot delete from an empty list")

        # Check for invalid position
        if n < 1:
            raise ValueError("Position must be 1 or greater")

        # Special case: deleting the head node
        if n == 1:
            self.he
            return

        current = self.head
        prev = None
```

```
            position = 1

            # Traverse to the nth node
            while current is not None and position < n:
                prev = current
                current = current.next
                position += 1

            # Check if position was out of range
            if current is None:
                raise ValueError(f"Position {n} is out of range")

            # Delete the node by skipping it
            prev.next = current.next

        def __len__(self):
            """Return the number of nodes in the list"""
            count = 0
            current = self.head
            while current is not None:
                count += 1
                current = current.next
            return count


    # Test the implementation
    if __name__ == "__main__":
        print("Testing Linked List Implementation\n")

        # Create a new linked list
        my_list = LinkedList()
        print("Initial list:")
        my_list.print_list()  # Should show empty list

        # Add some nodes
        print("\nAdding nodes with values 5, 10, 15, 20")
        for value in [5, 10, 15, 20]:
            my_list.add_node(value)
        my_list.print_list()  # Should show: 5 -> 10 -> 15 -> 20

        # Test deleting nodes
        print("\nDeleting node at position 2 (value 10):")
        my_list.delete_nth_node(2)
        my_list.print_list()  # Should show: 5 -> 15 -> 20

        print("\nDeleting first node (position 1, value 5):")
        my_list.delete_nth_node(1)
        my_list.print_list()  # Should show: 15 -> 20

        # Test error handling
        print("\nAttempting to delete from empty list:")
        empty_list = LinkedList()
        try:
            empty_list.delete_nth_node(1)
        except ValueError as e:
            print(f"Error caught: {e}")

        print("\nAttempting to delete at position 0:")
        try:
            my_list.delete_nth_node(0)
        except ValueError as e:
            print(f"Error caught: {e}")

        print("\nAttempting to delete at position 5 (out of range):")
        try:
            my_list.delete_nth_node(5)
        except ValueError as e:
            print(f"Error caught: {e}")
```

```
# Show final list
print("\nFinal state of the list:")
my_list.print_list()  # Should show: 15 -> 20
print(f"Number of nodes: {len(my_list)}")  # Should show 2
```

```
Testing Linked List Implementation

Initial list:
The list is empty

Adding nodes with values 5, 10, 15, 20
5 -> 10 -> 15 -> 20

Deleting node at position 2 (value 10):
5 -> 15 -> 20

Deleting first node (position 1, value 5):
15 -> 20

Attempting to delete from empty list:
Error caught: Cannot delete from an empty list

Attempting to delete at position 0:
Error caught: Position must be 1 or greater

Attempting to delete at position 5 (out of range):
Error caught: Position 5 is out of range

Final state of the list:
15 -> 20
Number of nodes: 2
```