66_Lavanya Kini

Assignment 03

```python
# Node class for linked list
class ListNode:
    def __init__(self, val=0, next=None):
        self.val = val
        self.next = next

# Linked List implementation
class LinkedList:
    def __init__(self):
        self.head = None

    def append(self, val):
        """Adds a new node to the end of the linked list"""
        if not self.head:
            self.head = ListNode(val)
            return
        current = self.head
        while current.next:
            current = current.next
        current.next = ListNode(val)

    def display(self):
        """Returns the list representation of the linked list"""
        result = []
        current = self.head
        while current:
            result.append(current.val)
            current = current.next
        return result

# Stack implementation using list
class Stack:
    def __init__(self):
        self.stack = []

    def push(self, val):
        """Pushes an element onto the stack"""
        self.stack.append(val)

    def pop(self):
        """Removes and returns the top element of the stack"""
        if not self.is_empty():
            return self.stack.pop()
        return None
    def peek(self):
        """Returns the top element without removing it"""
        if not self.is_empty():
            return self.stack[-1]
        return None

    def is_empty(self):
        """Checks if the stack is empty"""
        return len(self.stack) == 0

# Queue implementation using list
class Queue:
    def __init__(self):
        self.queue = []

    def enqueue(self, val):
        """Adds an element to the end of the queue"""
        self.queue.append(val)

    def dequeue(self):
        """Removes and returns the front element of the queue"""
        if not self.is_empty():
            return self.queue.pop(0)
        return None

    def front(self):
        """Returns the front element without removing it"""
        if not self.is_empty():
            return self.queue[0]
        return None

    def is_empty(self):
        """Checks if the queue is empty"""
        return len(self.queue) == 0

# Deque implementation using collections.deque
from collections import deque

class Deque:
    def __init__(self):
        self.deque = deque()

    def add_front(self, val):
        """Adds an element to the front of the deque"""
        self.deque.appendleft(val)

    def add_rear(self, val):
        """Adds an element to the rear of the deque"""
        self.deque.append(val)
```

```python
    def remove_front(self):
        """Removes and returns the front element of the deque"""
        if not self.is_empty():
            return self.deque.popleft()
        return None

    def remove_rear(self):
        """Removes and returns the rear element of the deque"""
        if not self.is_empty():
            return self.deque.pop()
        return None

    def is_empty(self):
        """Checks if the deque is empty"""
        return len(self.deque) == 0

# Example usage
if __name__ == "__main__":
    # Linked List example
    ll = LinkedList()
    ll.append(1)
    ll.append(2)
    ll.append(3)
    print("Linked List:", ll.display())

    # Stack example
    stack = Stack()
    stack.push(10)
    stack.push(20)
    stack.push(30)
    print("Stack Pop:", stack.pop())

    # Queue example
    queue = Queue()
    queue.enqueue(100)
    queue.enqueue(200)
    queue.enqueue(300)
    print("Queue Dequeue:", queue.dequeue())

    # Deque example
    deque_obj = Deque()
    deque_obj.add_front(5)
    deque_obj.add_rear(10)
    print("Deque Remove Front:", deque_obj.remove_front())
```

```
Linked List: [1, 2, 3]
Stack Pop: 30
Queue Dequeue: 100
Deque Remove Front: 5
```