```
class DequeUsingTwoStacks:
    def __init__(self):
        self.stack1 = []  # For operations on the "right" end
        self.stack2 = []  # For operations on the "left" end

    def push_to_left(self, value):
        while len(self.stack1) > 0:
            self.stack2.append(self.stack1.pop())  # Transfer all elements to
stack2
        self.stack2.append(value)  # Add the new value to the left
        while len(self.stack2) > 0:
            self.stack1.append(self.stack2.pop())  # Transfer all elements
back to stack1

    def push_to_right(self, value):
        self.stack1.append(value)  # Push directly to stack1

    def pop_from_left(self):
        if len(self.stack1) == 0:
            raise IndexError("Pop from empty deque")
        while len(self.stack1) > 0:
            self.stack2.append(self.stack1.pop())  # Transfer all elements to
stack2
        value = self.stack2.pop()  # Pop the leftmost element
        while len(self.stack2) > 0:
            self.stack1.append(self.stack2.pop())  # Transfer all elements
back to stack1
        return value

    def pop_from_right(self):
        if len(self.stack1) == 0:
            raise IndexError("Pop from empty deque")
        return self.stack1.pop()  # Pop directly from stack1


class DequeUsingStackQueue:
    def __init__(self):
        self.stack = []  # Stack for right-side operations
        self.queue = []  # List as a queue for left-side operations

    def push_to_left(self, value):
        self.queue.insert(0, value)  # Insert at the front of the queue

    def push_to_right(self, value):
        self.stack.append(value)  # Push to the stack

    def pop_from_left(self):
        if len(self.queue) > 0:
            return self.queue.pop(0)  # Remove from the front of the queue
        elif len(self.stack) > 0:
            while len(self.stack) > 0:
                self.queue.insert(0, self.stack.pop())  # Move all elements
to the queue
            return self.queue.pop(0)
        else:
            raise IndexError("Pop from empty deque")
```

```
    def pop_from_right(self):
        if len(self.stack) > 0:
            return self.stack.pop()  # Remove from the stack
        elif len(self.queue) > 0:
            while len(self.queue) > 0:
                self.stack.append(self.queue.pop(0))  # Move all elements to
the stack
            return self.stack.pop()
        else:
            raise IndexError("Pop from empty deque")


# Example usage of DequeUsingTwoStacks (Part A)
print("Part A: Deque Using Two Stacks\n")
deque1 = DequeUsingTwoStacks()
deque1.push_to_left(10)
deque1.push_to_right(20)
deque1.push_to_left(5)
print("Numbers in the deque for Part A:")
print("Pop from right: ", deque1.pop_from_right())  # Output: 20
print("Pop from left: ", deque1.pop_from_left())   # Output: 5
print("Pop from left: ", deque1.pop_from_left())   # Output: 10

# Example usage of DequeUsingStackQueue (Part B)
print("\nPart B: Deque Using Stack and Queue\n")
deque2 = DequeUsingStackQueue()
deque2.push_to_left(30)
deque2.push_to_right(40)
deque2.push_to_left(20)
print("Numbers in the deque for Part B:")
print("Pop from left: ", deque2.pop_from_left())   # Output: 20
print("Pop from right: ", deque2.pop_from_right())  # Output: 40
print("Pop from left: ", deque2.pop_from_left())   # Output: 30
```

```
C:\Python312\python.exe Z:\DSALGO-IDB2\termproject\Project1.py
Part A: Deque Using Two Stacks

Numbers in the deque for Part A:
Pop from right:  20
Pop from left:  5
Pop from left:  10

Part B: Deque Using Stack and Queue

Numbers in the deque for Part B:
Pop from left:  20
Pop from right:  40
Pop from left:  30

Process finished with exit code 0
```