```
class Node:
    def __init__(self, data):
        self.data = data
        self.next = None


class LinkedList:
    def __init__(self):
        self.head = None

    def print_list(self):
        current = self.head
        while current:
            print(current.data)
            current = current.next

    def append(self, data):
        new_Node = Node(data)

        if self.head is None:
            self.head = new_Node
            return

        last_node = self.head
        while last_node.next:
            last_node = last_node.next
        last_node.next = new_Node


class CircularLinkedList:
    def __init__(self):
        self.head = None

    def prepend(self, data):
        new_node = Node(data)
        cur = self.head
        new_node.next = self.head
        # print(cur.data)
        if not self.head:
            new_node.next = new_node
        else:
            # print(cur.next.data)
            # print(self.head.data)
            while cur.next != self.head:
                cur = cur.next
            cur.next = new_node
        self.head = new_node

    def append(self, data):
        if not self.head:
            self.head = Node(data)
            self.head.next = self.head
        else:
            new_node = Node(data)
            cur = self.head
            # print(cur)
            while cur.next != self.head:
                cur = cur.next
            cur.next = new_node
            new_node.next = self.head

    def print_list(self):
        cur = self.head
        while cur:
            print(cur.data)
            cur = cur.next
            if cur == self.head:
                break
```

```
    def remove(self, key):
        if self.head.data == key:
            cur = self.head
            while cur.next != self.head:
                cur = cur.next
            cur.next = self.head.next
            self.head = self.head.next
        else:
            cur = self.head
            prev = None
            while cur.next != self.head:
                prev = cur
                cur = cur.next
                if cur.data == key:
                    prev.next = cur.next
                    cur = cur.next

    def __len__(self):
        cur = self.head
        count = 0
        while cur:
            count += 1
            cur = cur.next
            if cur == self.head:
                break
        return count

    def split_list(self):
        size = len(self)

        if size == 0:
            return None
        if size == 1:
            return self.head

        mid = size // 2
        count = 0

        prev = None
        cur = self.head
        while cur and count < mid:
            count += 1
            prev = cur
            cur = cur.next
        prev.next = self.head

        split_clist = CircularLinkedList()
        while cur.next != self.head:
            split_clist.append(cur.data)
            cur = cur.next
        split_clist.append(cur.data)

        self.print_list()
        print("\n")
        split_clist.print_list()

    def remove_node(self, node):
        if self.head == node:
            cur = self.head
            while cur.next != self.head:
                cur = cur.next
            cur.next = self.head.next
            self.head = self.head.next
        else:
            cur = self.head
            prev = None
            while cur.next != self.head:
```

```
                prev = cur
                cur = cur.next
                if cur == node:
                    prev.next = cur.next
                    cur = cur.next

    def josephus_circle(self, step):
        cur = self.head
        while len(self) > 1:
            count = 1
            while count != step:
                cur = cur.next
                count += 1
            print("Removed: " + str(cur.data))
            self.remove_node(cur)
            cur = cur.next

    def is_circular_linked_list(self, input_list):
        cur = input_list.head
        while cur.next:
            cur = cur.next
            if cur.next == input_list.head:
                return True
        return False
```