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

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

    def addToEmpty(self, data):

        if self.last != None:
            return self.last

        newNode = Node(data)

        self.last = newNode

        self.last.next = self.last
        return self.last

    def addFront(self, data):

        if self.last == None:
            return self.addToEmpty(data)

        newNode = Node(data)

        newNode.next = self.last.next

        self.last.next = newNode

        return self.last

    def addEnd(self, data):
        if self.last == None:
            return self.addToEmpty(data)

        newNode = Node(data)

        newNode.next = self.last.next

        self.last.next = newNode

        self.last = newNode

        return self.last

    def addAfter(self, data, item):

        if self.last == None:
            return None

        newNode = Node(data)
        p = self.last.next
```

```
    while p:

        if p.data == item:

            newNode.next = p.next

            p.next = newNode

            if p == self.last:
                self.last = newNode
                return self.last
            else:
                return self.last
        p = p.next
        if p == self.last.next:
            print(item, "The given node is not present in the list")
            break

def deleteNode(self, last, key):

    if last == None:
        return

    if (last).data == key and (last).next == last:

        last = None

    temp = last
    d = None

    if (last).data == key:

        while temp.next != last:
            temp = temp.next
        temp.next = (last).next
        last = temp.next

    while temp.next != last and temp.next.data != key:
        temp = temp.next

    if temp.next.data == key:
        d = temp.next
        temp.next = d.next

    return last

def traverse(self):
    if self.last == None:
        print("The list is empty")
        return

    newNode = self.last.next
    while newNode:
        print(newNode.data, end=" ")
        newNode = newNode.next
```

```
            if newNode == self.last.next:
                break

if __name__ == "__main__":

    cll = CircularLinkedList()

    last = cll.addToEmpty(6)
    last = cll.addEnd(8)
    last = cll.addFront(2)
    last = cll.addAfter(10, 2)

    cll.traverse()

    last = cll.deleteNode(last, 8)
    print()
    cll.traverse()
```