```
class _Node:
    __slots__ = '_element', '_next'

    def __init__(self, element, next):
        self._element = element
        self._next = next

class CircularLinkedList:
    def __init__(self):
        self._head = None
        self._tail = None
        self._size = 0

    def __len__(self):
        return self._size

    def isempty(self):
        return self._size == 0

    def addlast(self, e):
        newest = _Node(e, None)
        if self.isempty():
            newest._next = newest
            self._head = newest
        else:
            newest._next = self._tail._next
            self._tail._next = newest
        self._tail = newest
        self._size += 1

    def display(self):
        p = self._head
        i = 0
        while i < len(self):
            print(p._element,end='-->')
            p = p._next
            i += 1
        print()

    def addfirst(self,e):
        newest = _Node(e, None)
        if self.isempty():
            newest._next = newest
            self._head = newest
            self._tail = newest
        else:
            self._tail._next = newest
            newest._next = self._head
            self._head = newest
        self._size += 1
```

```
def addany(self, e, position):
    newest = _Node(e, None)
    p = self._head
    i = 1
    while i < position - 1:
        p = p._next
        i = i + 1
    newest._next = p._next
    p._next = newest
    self._size += 1

def removefirst(self):
    if self.isempty():
        print('List is empty')
        return
    e = self._head._element
    self._tail._next = self._head._next
    self._head = self._head._next
    self._size -= 1
    if self.isempty():
        self._head = None
        self._tail = None
    return e

def removelast(self):
    if self.isempty():
        print('List is empty')
        return
    p = self._head
    i = 1
    while i < len(self) - 1:
        p = p._next
        i = i + 1
    self._tail = p
    p = p._next
    self._tail._next = self._head
    e = p._element
    self._size -= 1
    return e

def removeany(self, position):
    p = self._head
    i = 1
    while i < position - 1:
        p = p._next
        i = i + 1
    e = p._next._element
    p._next = p._next._next
    self._size -= 1
```

```
        return e

    def search(self,key):
        p = self._head
        index = 0
        while index < len(self):
            if p._element == key:
                return index
            p = p._next
            index = index + 1
        return -1

C = CircularLinkedList()
C.addlast(7)
C.addlast(4)
C.addlast(12)
C.display()
print('Size:',len(C))
C.addlast(8)
C.addlast(3)
C.display()
print('Size:',len(C))

C.addfirst(25)
C.display()
print('Size:',len(C))

C.addany(25,3)
C.display()
print('Size:',len(C))

ele = C.removefirst()
C.display()
print('Size:',len(C))
print('Removed Element:',ele)

ele = C.removelast()
C.display()
print('Size:',len(C))
print('Removed Element:',ele)

ele = C.removeany(3)
C.display()
print('Size:',len(C))
print('Removed Element:',ele)
```