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

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

class LinkedList:
    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():
            self._head = newest
        else:
            self._tail._next = newest
        self._tail = newest
        self._size += 1

    def addfirst(self, e):
        newest = _Node(e, None)
        if self.isempty():
            self._head = newest
            self._tail = newest
        else:
            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._head = self._head._next
        self._size -= 1
        if self.isempty():
            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
        e = p._element
        self._tail._next = None
        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 display(self):
        p = self._head
        while p:
            print(p._element,end='-->')
            p = p._next
        print()

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

L = LinkedList()
L.addlast(7)
L.addlast(4)
L.addlast(12)
L.addlast(8)
L.addlast(3)
L.display()
i = L.search(8)
print('Result:',i)
index = L.search(20)
print('Result:',i)

L.addfirst(25)
L.display()
print('Size:',len(L))
L.addlast(35)
L.display()
print('Size:',len(L))

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

print('Size:',len(L))
ele = L.removeany(3)
L.display()
print('Size:',len(L))

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