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

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

class DoublyLinkedList:
    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, None)
        if self.isempty():
            self._head = newest
            self._tail = newest
        else:
            self._tail._next = newest
            newest._prev = self._tail
            self._tail = newest
        self._size += 1

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

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

```
        p._next._prev = newest
        p._next = newest
        newest._prev = p
        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
        else:
            self._head._prev = None
        return e

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

    def removeany(self, position):
        if self.isempty():
            print('List is empty')
            return
        p = self._head
        i = 1
        while i < position - 1:
            p = p._next
            i = i + 1
        e = p._next._element
        p._next = p._next._next
        p._next._prev = p
        self._size -= 1
        return e

    def display(self):
        p = self._head
        while p:
            print(p._element,end=' --> ')
            p = p._next
        print()
```

```
    def displayrev(self):
        p = self._tail
        while p:
            print(p._element,end=' <-- ')
            p = p._prev
        print()

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

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

L.addany(20,3)
L.display()
print('Size:',len(L))
L.displayrev()

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

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

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