```
from exceptions import Empty

class DoublyLinkedList:
    class _Node:
        __slots__ ='_element','_prev','_next'
        def __init__(self,element,prev,next):
            self._element = element
            self._prev = prev
            self._next = next
    def __init__(self):
        self._head = self._Node(None,None,None)
        self._tail = self._Node(None,None,None)
        self._head._next = self._tail
        self._tail._prev = self._head
        self._size = 0
    def __len__(self):
        return self._size
    def is_empty(self):
        return self._size == 0

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

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

    def add_any(self,e,pos):
        newest = self._Node(e,None, None)
        thead = self._head
        i = 1
        while i < pos:
            thead = thead._next
            i += 1
        newest._next = thead._next
        thead._next = newest
        thead._next._prev = newest
        newest._prev = thead
        self._size += 1

    def remove_first(self):
        if self.is_empty():
            raise Empty('List is Empty')
        value = self._head._element
        self._head = self._head._next
        self._head._prev = None
        self._size -= 1
        if self.is_empty():
            self._tail = None
        return value

    def remove_last(self):
        if self.is_empty():
            raise Empty('List is Empty')
        thead = self._head
        i = 0
        while i < len(self)-2:
            thead = thead._next
            i += 1
        self._tail = thead
        thead = thead._next
        value = thead._element
        self._tail._next = None
        self._size -= 1
        return value

    def remove_any(self, pos):
        if self.is_empty():
            raise Empty('List is Empty')
        thead = self._head
        i = 1
        while i < pos-1:
            thead = thead._next
            i += 1
        thead._next = thead._next._next
        thead._next._next._prev = thead
        self._size -= 1

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


L = DoublyLinkedList()
L.add_last(10)
L.add_last(20)
L.add_last(30)
L.add_last(40)
L.display()
print('Delete: ',L.remove_first())
L.display()
L.add_first(70)
L.display()
print('Delete: ',L.remove_last())
L.display()
L.add_any(100,2)
L.display()
L.remove_any(2)
L.display()
```