

class Node

data-type num

Node next

Node(data-type n)

assign n to num

assign null/None to next

class SLinkedList

Node start // beginning of list (first element if not null/None)

Node end // end of list (last element if not null/None)

int size // number of Node elements in list

SLinkedList() // constructor

assign null/None to start // empty list

assign null/None to end // empty list

assign 0 to size // no elements

// add element to beginning of of the list

void addFirst(data-type num)

create new Node n with value of num

new Node n if (start == null/None) // no elements so make start and end point to

assign n to start

assign n to end

else

set n.next to start // connects n to element at start

set start.prev to n // connects element at start to n

set start to n // makes n the first element

increment size by 1

// add element to the end of the list

void addLast(data-type num)

create new Node n with value of num

new Node n if (start == null/None) // no elements so make start and end point to

assign n to start

assign n to end

else

set end.next to n // connects last element at end to new

element n

set end to n // makes n the last element

increment size by 1

// delete first element and return its value

data-type deleteFirst()

```

        if (start == null/None) // no elements
            throw exception with message that list is empty

        Node n = start // save the reference to the first node so can return

        if (start == end) // one element so make start/end null/None since
deleting
            assign null/None to start
            assign null/None to end
        else
            set start to start.next // setting to next element
            set start.prev to null/None since now first element
            decrement size by 1
            return n.num
    }

    // delete last element and return its value
    data-type deleteLast()

        if (start == null/None) // no elements
            throw exception with message that list is empty

        Node n = end // save the reference to the last node so can return

        if (start == end) // one element so make start/end null/None since
deleting
            assign null/None to start
            assign null/None to end
        else
            Node tmp = start
            Loop until tmp.next == end // next to last element
            set end to tmp // now end is at next to last element
            set end.next to null/None since now last element
            decrement size by 1
            return n.num

    // return the number of nodes in list
    public int count()
        return size

    // print values of all elements from first to last
    void printNextList()
        Node n = start // start with first element
        while (n != null) // traverse all elements
            print n.num
            n=n.next

    // return value of start
    Node getStart()
        return start

    // return value of end
    Node getEnd()

```

return end