class Node:

    def \_\_init\_\_(self, e, n, p):

        self.element = e

        self.next = n

        self.prev = p

class DoublyList:

    def \_\_init\_\_(self, a):

        if type(a) == list:

            self.head = None

            self.tail = None

            x = len(a) - 1

            f = Node(None, None, None)

            for i in range(0, len(a)):

                n = Node(a[x], self.head, None)

                x -= 1

                if n.next == None:

                    tail = n

                    self.head = n

                    self.tail = n

                else:

                    self.head = n

                    tail.prev = n

                    tail = n

            f.next = self.head

            f.prev = self.tail

            self.head.prev = f

            self.tail.next = f

            self.head = f

    def countNode(self):

        f = self.head.next

        count = 0

        while f != self.head:

            count += 1

            f = f.next

        return count

    def forwardprint(self):

        f = self.head.next

        x = ""

        while f != self.head:

            if f.next != self.head:

                x += str(f.element) + ", "

            else:

                x += str(f.element) + "."

            f = f.next

        print(x)

    def backwardprint(self):

        f = self.tail

        x = ""

        while f != self.head:

            if f.prev != self.head:

                x += str(f.element) + ", "

            else:

                x += str(f.element) + "."

            f = f.prev

        print(x)

    def nodeAt(self, idx):

        f = self.head.next

        if idx >= 0 and idx < self.countNode():

            for i in range(0, idx):

                h = f.next

            return f

        else:

            return Node("Index Error", None, None)

    def indexOf(self, elmnt):

        f = self.head.next

        count = 0

        while f != self.head:

            if f.element == elmnt:

                return count

            f = f.next

            count += 1

        return -1

    def insert(self, elmnt, idx):

        f = self.head.next

        if idx == 0:

            x = self.head

            y = Node(elmnt, f, x)

            f1 = self.nodeAt(0)

            self.head.next = y

            f1.prev =y

        elif idx > 0 and idx < self.countNode():

            f1 = self.nodeAt(idx-1)

            f2 = self.nodeAt(idx)

            y = Node(elmnt,f1.next, f1)

            f1.next = y

            f2.prev = y

        elif idx == self.countNode():

            f1 = self.nodeAt(idx-1)

            y = Node(elmnt, None, None)

            f1.next = y

            y.next = self.head

            y.prev = f1

            self.tail = y

    def remove(self, idx):

        f = self.head.next

        if idx == 0:

            elmnt = f.element

            self.head.next = f.next

            f.next.prev = self.head

            f.next = None

            f.prev = None

            return str(elmnt)

        elif idx == self.countNode() - 1:

            fidx = self.nodeAt(idx)

            f1 = self.nodeAt(idx-1)

            elmnt = nidx.element

            f1.next = self.head

            fidx.next = None

            fidx.prev = None

            self.tail = f1

            return str(elmnt)

        elif idx > 0 and idx < self.countNode():

            fidx = self.nodeAt(idx)

            f1 = self.nodeAt(idx-1)

            f2 = self.nodeAt(idx+1)

            elmnt = fidx.element

            f1.next = fidx.next

            f2.prev = fidx.prev

            fidx.next = None

            fidx.prev = None

            return str(elmnt)

        else:

            return None

print("///  Test 01  ///")

a1 = [10, 20, 30, 40]

h1 = DoublyList(a1)

h1.forwardprint()

h1.backwardprint()

print(h1.countNode())

print("///  Test 02  ///")

myNode = h1.nodeAt(2)

print(myNode.element)

print("///  Test 03  ///")

index = h1.indexOf(40)

print(index)

print("///  Test 04  ///")

a2 = [10, 20, 30, 40]

h2 = DoublyList(a2)

h2.forwardprint()

h2.insert(85,0)

h2.forwardprint()

h2.backwardprint()

print()

h2.insert(95,3)

h2.forwardprint()

h2.backwardprint()

print()

h2.insert(75,6)

h2.forwardprint()

h2.backwardprint()

print("///  Test 05  ///")

a3 = [10, 20, 30, 40, 50, 60, 70]

h3 = DoublyList(a3)

h3.forwardprint()

print("Removed element: "+ h3.remove(0))

h3.forwardprint()

h3.backwardprint()

print("Removed element: "+ h3.remove(3))

h3.forwardprint()

h3.backwardprint()

print("Removed element: "+ h3.remove(4))

h3.forwardprint()

h3.backwardprint()