class Node:

  def \_\_init\_\_(self, e, n):

    self.element = e

    self.next = n

class LinkedList:

  def \_\_init\_\_(self, a):

  #  Design the constructor based on data type of a. If 'a' is built in python list then

  #  Creates a linked list using the values from the given array. head will refer

  #  to the Node that contains the element from a[0]

  #  Else Sets the value of head. head will refer

  # to the given LinkedList

  # Hint: Use the type() function to determine the data type of a

    if type(a) is list:

      self.head = Node(a[0],None)

      few=self.head

      for i in range(1,len(a)):

        new=Node(a[i],None)

        few.next=new

        few=new

  # Count the number of nodes in the list

  def countNode(self):

    # To Do

    new=self.head

    count=0

    while new!=None:

      count+=1

      new=new.next

    return count

  # Print elements in the list

  def printList(self):

    # To Do

    x=''

    new=self.head

    while new!=None:

      x+=str(new.element)+','

      new=new.next

    x=x[0:-1]

    print(x)

  # returns the reference of the Node at the given index. For invalid index return None.

  def nodeAt(self, idx):

    # To Do

    count=0

    new=self.head

    while new!=None:

      if count==idx:

        return new

      count+=1

      new=new.next

    return None

  # returns the element of the Node at the given index. For invalid idx return None.

  def get(self, idx):

    # To Do

    count=0

    new=self.head

    while new!=None:

      if count==idx:

        return new.element

      count+=1

      new=new.next

    return None

  # updates the element of the Node at the given index.

  # Returns the old element that was replaced. For invalid index return None.

  # parameter: index, element

  def set(self, idx, elem):

    # To Do

    count=0

    new=self.head

    while new!=None:

      if count==idx:

        x=new.element

        new.element=elem

        return x

      count+=1

      new=new.next

    return None

  # returns the index of the Node containing the given element.

  # if the element does not exist in the List, return -1.

  def indexOf(self, elem):

    # To Do

    count=0

    new=self.head

    x=True

    while new!=None:

      if new.element==elem:

        x=False

        return count

      count+=1

      new=new.next

    return -1

  # returns true if the element exists in the List, return false otherwise.

  def contains(self, elem):

    # To Do

    count=0

    new=self.head

    x=True

    while new!=None:

      if new.element==elem:

        x=False

        return True

      count+=1

      new=new.next

    return False

  # Makes a duplicate copy of the given List. Returns the reference of the duplicate list.

  def copyList(self):

    # To Do

    count=0

    new=self.head

    while new!=None:

      count+=1

      new=new.next

    x=[0]\*count

    new=self.head

    count=0

    while new!=None:

      x[count]=new.element

      count+=1

      new=new.next

    return x

  # Makes a reversed copy of the given List. Returns the head reference of the reversed list.

  def reverseList(self):

    # To Do

    count=0

    new=self.head

    while new!=None:

      count+=1

      new=new.next

    x=[0]\*count

    new=self.head

    while new!=None:

      count-=1

      x[count]=new.element

      new=new.next

    return x

  # inserts Node containing the given element at the given index

  # Check validity of index.

  def insert(self, elem, idx):

    # To Do

    if idx==0:

      n=Node(elem,self.head)

      self.head=n

    else:

      new=self.head

      count=0

      while new.next!=None:

        if count==idx:

          n=Node(elem,None)

          n.next=new

          temp.next=n

        count+=1

        temp=new

        new=new.next

      if idx>count:

        n=Node(elem,None)

        new.next=n

  # removes Node at the given index. returns element of the removed node.

  # Check validity of index. return None if index is invalid.

  def remove(self, idx):

    # To Do

    if idx==0:

      f=self.head

      self.head=self.head.next

      return f.element

    else:

      new=self.head

      count=0

      while new!=None:

        if count==idx:

          temp.next=new.next

          return new.element

        count+=1

        temp=new

        new=new.next

    return None

  # Rotates the list to the left by 1 position.

  def rotateLeft(self):

    # To Do

    count=0

    f=self.head

    self.head=self.head.next

    new=self.head

    while new!=None:

      if new.next==None:

        new.next=f

        f.next=None

      count+=1

      new=new.next

  # Rotates the list to the right by 1 position.

  def rotateRight(self):

    # To Do

    count=0

    new=self.head

    while new!=None:

      if new.next==None:

        temp.next=None

        new.next=self.head

        self.head=new

        break

      count+=1

      temp=new

      new=new.next

print("////// Test 01 //////")

a1 = [10, 20, 30, 40]

h1 = LinkedList(a1) # Creates a linked list using the values from the array

# head will refer to the Node that contains the element from a[0]

h1.printList() # This should print: 10,20,30,40

print(h1.countNode()) # This should print: 4

print("////// Test 02 //////")

# returns the reference of the Node at the given index. For invalid idx return None.

myNode = h1.nodeAt(1)

print(myNode.element) # This should print: 20. In case of invalid index This will generate an Error.

print("////// Test 03 //////")

# returns the element of the Node at the given index. For invalid idx return None.

val = h1.get(2)

print(val) # This should print: 30. In case of invalid index This will print None.

print("////// Test 04 //////")

# updates the element of the Node at the given index.

# Returns the old element that was replaced. For invalid index return None.

# parameter: index, element

print(h1.set(1,85)) # This should print: 20

h1.printList() # This should print: 10,85,30,40.

print(h1.set(15,85)) # This should print: None

h1.printList() # This should print: 10,85,30,40.

print("////// Test 05 //////")

# returns the index of the Node containing the given element.

# if the element does not exist in the List, return -1.

index = h1.indexOf(40)

print(index) # This should print: 3. In case of element that doesn't exists in the list this will print -1.

print("////// Test 06 //////")

# returns true if the element exists in the List, return false otherwise.

ask = h1.contains(40)

print(ask) # This should print: True.

print("////// Test 07 //////")

a2 = [10,20,30,40,50,60,70]

h2 = LinkedList(a2) # uses theconstructor where a is an built in list

h2.printList() # This should print: 10,20,30,40,50,60,70.

# Makes a duplicate copy of the given List. Returns the head reference of the duplicate list.

copyH=h2.copyList() # Head node reference of the duplicate list

h3 = LinkedList(copyH) # uses the constructor where a is head of a linkedlist

h3.printList() # This should print: 10,20,30,40,50,60,70.

print("////// Test 08 //////")

a4 = [10,20,30,40,50]

h4 = LinkedList(a4) # uses theconstructor where a is an built in list

h4.printList() # This should print: 10,20,30,40,50.

# Makes a reversed copy of the given List. Returns the head reference of the reversed list.

revH=h4.reverseList() # Head node reference of the reversed list

h5 = LinkedList(revH) # uses the constructor where a is head of a linkedlist

h5.printList() # This should print: 50,40,30,20,10.

print("////// Test 09 //////")

a6 = [10,20,30,40]

h6 = LinkedList(a6) # uses theconstructor where a is an built in list

h6.printList() # This should print: 10,20,30,40.

# inserts Node containing the given element at the given index. Check validity of index.

h6.insert(85,0)

h6.printList() # This should print: 85,10,20,30,40.

h6.insert(95,3)

h6.printList() # This should print: 85,10,20,95,30,40.

h6.insert(75,6)

h6.printList() # This should print: 85,10,20,95,30,40,75.

print("////// Test 10 //////")

a7 = [10,20,30,40,50,60,70]

h7 = LinkedList(a7) # uses theconstructor where a is an built in list

h7.printList() # This should print: 10,20,30,40,50,60,70.

# removes Node at the given index. returns element of the removed node.

# Check validity of index. return None if index is invalid.

print("Removed element:",h7.remove(0)) # This should print: Removed element: 10

h7.printList() # This should print: 20,30,40,50,60,70.

print("Removed element: ",h7.remove(3)) # This should print: Removed element: 50

h7.printList() # This should print: 20,30,40,60,70.

print("Removed element: ",h7.remove(4)) # This should print: Removed element: 70

h7.printList() # This should print: 20,30,40,60.

print("////// Test 11 //////")

a8 = [10,20,30,40]

h8 = LinkedList(a8) # uses theconstructor where a is an built in list

h8.printList() # This should print: 10,20,30,40.

# Rotates the list to the left by 1 position.

h8.rotateLeft()

h8.printList() # This should print: 20,30,40,10.

print("////// Test 12 //////")

a9 = [10,20,30,40]

h9 = LinkedList(a9) # uses theconstructor where a is an built in list

h9.printList() # This should print: 10,20,30,40.

# Rotates the list to the right by 1 position.

h9.rotateRight()

h9.printList() # This should print: 40,10,20,30.