class Node:

def \_\_init\_\_(self, e, n):

self.element = e

self.next = n

class LinkedList:

def \_\_init\_\_(self, a):

if type(a)==list:

self.head = Node(a[0],None)

tail = self.head

for i in range(1,len(a)):

n = Node(a[i], None)

tail.next=n

tail=n

elif a==None:

self.head=None

else:

temp=a.head

self.head = Node(temp.element,None)

tail=self.head

temp=temp.next

while temp!=None:

n=Node(temp.element,None)

tail.next=n

tail=n

temp=temp.next

# Count the number of nodes in the list

def countNode(self):

temp=self.head

count=0

while temp!=None:

count+=1

temp=temp.next

return count

# Print elements in the list

def printList(self):

count=self.countNode()

temp=self.head

c=0

while temp!=None:

if c!=count-1:

print(temp.element,end=", ")

else:

print(temp.element)

temp=temp.next

c+=1

print()

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

def nodeAt(self, idx):

temp=self.head

c=0

length=self.countNode()

if length>idx:

while temp!=None:

if c==idx:

return temp

else:

c+=1

temp=temp.next

else:

return Node(None,None)

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

def get(self, idx):

temp=self.head

c=0

length=self.countNode()

if length>idx:

while temp!=None:

if c==idx:

return temp.element

else:

c+=1

temp=temp.next

else:

return Node(None,None)

def indexOf(self, elem):

temp=self.head

c=0

flag=True

while temp!=None:

if temp.element==elem:

flag=True

return c

else:

flag=False

c+=1

temp=temp.next

if flag==False:

return -1

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

def contains(self, elem):

flag=True

temp=self.head

c=0

while temp!=None:

if temp.element==elem:

flag= True

break

else:

c+=1

temp=temp.next

flag=False

return flag

def set(self, idx, elem):

len=self.countNode()

if len<idx:

return None

else:

n=self.nodeAt(idx)

temp=n.element

n.element=elem

return temp

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

def copyList(self):

temp=self.head

dup= Node(temp.element,None)

tail=dup

temp=temp.next #for adding the second temp.elem

while temp!=None:

n=Node(temp.element,None)

tail.next=n

tail=n

temp=temp.next

copylist=LinkedList(None)

copylist.head=dup

return copylist

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

def reverseList(self): #10,20,30,40,50

temp=self.head

reverse=Node(temp.element,None)

temp=temp.next

while temp!=None:

n=Node(temp.element,None)

n.next=reverse #connecting the previous node

reverse=n

temp=temp.next

new=LinkedList(None)

new.head=reverse

return new

# inserts Node containing the given element at the given index

# Check validity of index.

def insert(self, elem, idx):

length=self.countNode()

if length<idx:

return "Invalid index"

else:

n=Node(elem, None)

if idx==0:

n.next=self.head

self.head=n

elif idx==length:

n1=self.nodeAt(idx-1)

n1.next=n

n.next=None

else:

n1=self.nodeAt(idx-1)

n2=self.nodeAt(idx)

n.next=n2

n1.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):

len=self.countNode()

if idx>len-1:

return None

else:

if idx==0:

val = self.head.element

self.head = self.head.next

return val

elif idx==len-1:

remove=self.nodeAt(idx)

val=remove.element

n1=self.nodeAt(idx-1)

n1.next=None

return val

else:

remove=self.nodeAt(idx)

val=remove.element

n1=self.nodeAt(idx-1)

n2=self.nodeAt(idx+1)

n1.next=n2

return val

# Rotates the list to the left by 1 position.

def rotateLeft(self):

temp=self.head

count=self.countNode()

c=0

while temp!=None:

if c==count-1:

break

temp=temp.next

c+=1

temp.next=self.head

newhead=self.head.next #adding the second location as newhead

self.head.next=None #forgetting old self.head

self.head=newhead

return self.head

# Rotates the list to the right by 1 position.

def rotateRight(self):

last=(self.countNode())-1

n1=self.nodeAt(last-1)

n2= n1.next

n1.next=None

n2.next=self.head

self.head=n2

return self.head

#.....................................................

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.