#Task\_01

class Node:

def \_\_init\_\_(self,value,pre\_node=None,next\_node=None):

self.value=value

self.pre\_node=pre\_node

self.next\_node=next\_node

def printNode(self):

if self.value!=None:

print(self.value,end=' <--> ')

else:

print(self.value)

class DoublyList:

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

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

self.head.next\_node = self.head.pre\_node = self.head

for i in range(len(a)):

newNode = Node(a[i], None, None)

newNode.next\_node=self.head

newNode.pre\_node=self.head.pre\_node

self.head.pre\_node.next\_node=newNode

self.head.pre\_node=newNode

else:

last\_node=self.head.pre\_node

newNode.next\_node=self.head

newNode.pre\_node=last\_node

last\_next\_node=newNode

#-------------------------------------------------------------------------

#Task\_02

class Node:

def \_\_init\_\_(self,value,pre\_node=None,next\_node=None):

self.value=value

self.pre\_node=pre\_node

self.next\_node=next\_node

def printNode(self):

print(self.value,end=' ')

#----------------(a)---------------------

#---------------01------------------------

class DoublyList:

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

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

self.head.next\_node = self.head.pre\_node = self.head

for i in range(len(a)):

newNode = Node(a[i], None, None)

newNode.next\_node=self.head

newNode.pre\_node=self.head.pre\_node

self.head.pre\_node.next\_node=newNode

self.head.pre\_node=newNode

else:

last\_node=self.head.pre\_node

newNode.next\_node=self.head

newNode.pre\_node=last\_node

last\_next\_node=newNode

#---------------02------------------------

def showList(self):

n=self.head.next\_node

if n.value is not None:

while n!=self.head:

n.printNode()

n=n.next\_node

else:

print('Empty list')

def node\_At(self,value):

n=self.head.next\_node

for x in range(0,value):

n=n.next\_node

return n

#---------------03------------------------

def insert(self, newElement) :

n = Node(newElement, None, None)

n.next\_node = self.head.next\_node

n.pre\_node = self.head

self.head.next\_node = n

n.next\_node.pre\_node = n

#---------------04------------------------

def insert\_at(self, newElement, index):

n = self.head.next\_node

while (n != self.head):

if (n.value == newElement):

print("Same Element")

return

n=n.next\_node

element=self.node\_At(index-1)

n1=Node(newElement,None,None)

n1.next\_node=element.next\_node

element.next\_node.pre\_node=n1

element.next\_node = n1

n1.pre\_node=element

#---------------05-----------------------

def remove(self,index):

if(index>0):

index=self.node\_At(index)

pre\_node = index.pre\_node

q = index.next\_node

pre\_node.next\_node = q

q.pre\_node = pre\_node

index.next\_node = index.pre\_node = None

index.value = None

else:

print("Error")

#---------------06------------------------

def removeKey(self, deletekey):

n=self.head.next\_node

while(n!=self.head):

if (n.value!= deletekey):

n=n.next\_node

else:

pre\_node=n.pre\_node

q=n.next\_node

pre\_node.next\_node=q

q.pre\_node=pre\_node

n=n.next\_node

dbl=DoublyList([10,20,30,40,50])

dbl.showList()

print()

print('showList')

print()

dbl.insert(90)

dbl.showList()

print()

print("insert")

print()

dbl.insert\_at(80,2)

dbl.showList()

print()

print('insert\_at')

print()

dbl.remove(4)

dbl.showList()

print()

print('remove')

print()

dbl.removeKey(2)

dbl.showList()

print()

print('removeKey')

Results:

#02

10 20 30 40 50

showList

#03

90 10 20 30 40 50

insert

#04

90 10 80 20 30 40 50

insert\_at

#05

90 10 80 20 40 50

remove

#06

90 10 80 20 40 50

removeKey