class node:

data = None

next = None

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

self.data = data

class LinklistFunction:

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

self.head = None

self.tail = None

def createLlist(self):

self.head

datas = int(input("Enter the data : "))

while(datas != -1):

if(self.head == None):

self.head = node(datas)

self.tail = self.head

else:

newNode = node(datas)

self.tail.next = newNode

self.tail = self.tail.next

datas = int(input("Enter the data : "))

return self.head

def \_\_display(self, tempHead):

if(tempHead == None):

return

print(tempHead.data," ", end=" ")

self.\_\_display(tempHead.next)

def display(self):

if(head == None):

tempHead = self.head

else:

tempHead = head

self.\_\_display(tempHead)

def deletehead(self):

if(self.head != None):

temp = self.head

self.head = self.head.next

temp.next = None

del temp

return self.head

print("LinkList is Empty")

def deleteIndexNode(self, index):

if(self.head != None):

temp = self.head

p = self.head

count = 1

if(index == 1):

self.head = self.deletehead()

return

while(count != index-1):

if(count == index-1 and temp.next.next == None):

del temp.next

temp.next = None

return

if(temp.next != None):

temp = temp.next

count+=1

else:

print('Your index is invalid ')

return

if(temp.next.next != None):

p = temp.next.next

del temp.next

temp.next = p

else:

del temp.next

temp.next = None

else:

print("linklist is empty")

return -1

def searchingNode(self, data):

if(self.head != None):

temp = self.head

count = 1

while(temp != None):

if(temp.data == data):

return count

count+=1

temp = temp.next

print("This Element does Not Exist in linkList")

return -1

def insertAtFirst(self, data):

if(data != -1):

ob = node(data)

if(self.head == None):

self.head = ob

return

else:

ob.next = self.head

self.head = ob

return

return -1

def getLength(self):

if(self.head == None):

return 0

temp = self.head

count = 1

while(temp.next != None):

temp = temp.next

count+=1

return count

def insertionAtIndedx(self, index, data):

if(index == 1):

self.insertAtFirst(data)

return

if(index <= self.getLength()+1):

newNode = node(data)

count = 1

temp = self.head

while(count != index-1):

temp = temp.next

count+=1

if(index == self.getLength()+1):

temp.next = newNode

return

newNode.next = temp.next

temp.next = newNode

return

print("Incorrect index")

return -1

def getMid(self):

if self.head == None:

return 0

slow=self.head

fast=self.head.next

count = 1

while(fast != None and fast.next != None):

slow = slow.next

count+=1

if(fast.next.next == None):

fast = fast.next

else:

fast = fast.next.next

return slow.data