**Lab 9**

Name: Omerullah Ansari

ID: 65584

Q1

def reverseString(s):

if len(s) <= 1:

return s

else:

return reverseString(s[1:]) + s[0]

# Test the function

i = "hello"

reversedString = reverseString(i)

print("Original string:", i)

print("Reversed string:", reversedString)



Q2

class Node:

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

self.data = data

self.next = None

class LinkedList:

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

self.head = None

def insert(self, data):

new\_node = Node(data)

if self.head is None:

self.head = new\_node

else:

current = self.head

while current.next:

current = current.next

current.next = new\_node

def deleteKthNode(self, k):

if k <= 0 or self.head is None:

return

self.deleteKthNodeRecursive(self.head, k)

def deleteKthNodeRecursive(self, node, k):

if node is None:

return None

if k == 1:

temp = node.next

del node

return temp

node.next = self.deleteKthNodeRecursive(node.next, k - 1)

return node

def display(self):

current = self.head

while current:

print(current.data, end=" ")

current = current.next

print("None")

ll = LinkedList()

ll.insert(1)

ll.insert(2)

ll.insert(3)

ll.insert(4)

ll.insert(5)

print("Original linked list:")

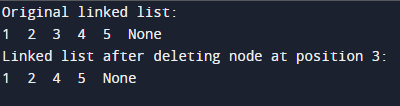
ll.display()

k = 3

ll.deleteKthNode(k)

print(f"Linked list after deleting node at position {k}:")

ll.display()



Q3

def mergeSort(stack):

if len(stack) > 1:

mid = len(stack) // 2

left\_half = stack[:mid]

right\_half = stack[mid:]

mergeSort(left\_half)

mergeSort(right\_half)

merge(stack, left\_half, right\_half)

def merge(stack, left\_half, right\_half):

i = j = k = 0

while i < len(left\_half) and j < len(right\_half):

if left\_half[i] < right\_half[j]:

stack[k] = left\_half[i]

i += 1

else:

stack[k] = right\_half[j]

j += 1

k += 1

while i < len(left\_half):

stack[k] = left\_half[i]

i += 1

k += 1

while j < len(right\_half):

stack[k] = right\_half[j]

j += 1

k += 1

def sortStack(stack):

if stack:

temp = stack.pop()

sortStack(stack)

insertSorted(stack, temp)

def insertSorted(stack, element):

if not stack or stack[-1] <= element:

stack.append(element)

else:

temp = stack.pop()

insertSorted(stack, element)

stack.append(temp)

stack = [5, 2, 7, 3, 1]

print("Original stack:", stack)

sortStack(stack)

print("Sorted stack:", stack)

