**EXP 2: Implementation of Singly linked list**

class Node:

"""Class to represent each node in the linked list."""

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

self.data = data

self.next = None

class SinglyLinkedList:

"""Singly linked list class with basic operations."""

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

self.head = None

def insert\_at\_beginning(self, data):

"""Insert a node at the beginning of the list."""

new\_node = Node(data)

new\_node.next = self.head

self.head = new\_node

def insert\_at\_end(self, data):

"""Insert a node at the end of the list."""

new\_node = Node(data)

if self.head is None:

self.head = new\_node

return

current = self.head

while current.next:

current = current.next

current.next = new\_node

def insert\_at\_position(self, data, position):

"""Insert a node at a specific position in the list."""

new\_node = Node(data)

if position == 1:

new\_node.next = self.head

self.head = new\_node

return

current = self.head

current\_position = 1

while current and current\_position < position - 1:

current = current.next

current\_position += 1

if not current:

print(f"Position {position} is out of bounds.")

return

new\_node.next = current.next

current.next = new\_node

def delete\_from\_beginning(self):

"""Delete a node from the beginning of the list."""

if self.head is None:

print("List is empty, nothing to delete.")

return

self.head = self.head.next

def delete\_from\_end(self):

"""Delete a node from the end of the list."""

if self.head is None:

print("List is empty, nothing to delete.")

return

if self.head.next is None:

self.head = None

return

current = self.head

while current.next.next:

current = current.next

current.next = None

def delete\_from\_position(self, position):

"""Delete a node from a specific position in the list."""

if self.head is None:

print("List is empty, nothing to delete.")

return

if position == 1:

self.head = self.head.next

return

current = self.head

current\_position = 1

while current and current\_position < position - 1:

current = current.next

current\_position += 1

if not current or not current.next:

print(f"Position {position} is out of bounds.")

return

current.next = current.next.next

def traverse(self):

"""Traverse and print the list elements."""

if self.head is None:

print("List is empty.")

return

current = self.head

while current:

print(current.data, end=" -> ")

current = current.next

print("None")

# Example Usage

if \_\_name\_\_ == "\_\_main\_\_":

sll = SinglyLinkedList()

# Insertions

sll.insert\_at\_beginning(10)

sll.insert\_at\_end(20)

sll.insert\_at\_position(15, 2)

sll.insert\_at\_position(5, 1)

sll.insert\_at\_end(25)

sll.traverse() # Output: 5 -> 10 -> 15 -> 20 -> 25 -> None

# Deletions

sll.delete\_from\_beginning()

sll.traverse() # Output: 10 -> 15 -> 20 -> 25 -> None

sll.delete\_from\_end()

sll.traverse() # Output: 10 -> 15 -> 20 -> None

sll.delete\_from\_position(2)

sll.traverse() # Output: 10 -> 20 -> None