Syit_380_umang

1)Implement Linked List. Write the code to concatenate following linked lists. A = [1,2,3,4] B= ['a','b','c']

Code:

```
class Node:
    def __init__(self, element, next=None):
        self.element = element
        self.next = next


class LinkedList:

    def __init__(self):
        self.head = None
        self.size = 0

    def display(self):
        if self.size == 0:
            print("No element")
            return
        first = self.head
        print(first.element)
        first = first.next
        while first:
            print(first.element)
            first = first.next

    def add_head(self, e):
        temp = self.head
        self.head = Node(e)
        self.head.next = temp
        self.size += 1
```

```
    def get_tail(self):
        last_object = self.head
        while last_object.next is not None:
            last_object = last_object.next
        return last_object

    def add_tail(self, e):
        new_value = Node(e)
        self.get_tail().next = new_value
        self.size += 1

    def get_node_at(self, index):
        element_node = self.head
        counter = 0
        if index > self.size - 1:
            print("Index out of bound")
            return None
        while counter < index:
            element_node = element_node.next
            counter += 1
        return element_node

    def add_between_list(self, position, element):
        if position > self.size:
            print("Index out of bound")
        elif position == self.size:
            self.add_tail(element)
```

Syit_380_umang

```
    def get_tail(self):
        last_object = self.head
        while last_object.next is not None:
            last_object = last_object.next
        return last_object

    def add_tail(self, e):
        new_value = Node(e)
        self.get_tail().next = new_value
        self.size += 1

    def get_node_at(self, index):
        element_node = self.head
        counter = 0
        if index > self.size - 1:
            print("Index out of bound")
            return None
        while counter < index:
            element_node = element_node.next
            counter += 1
        return element_node

    def add_between_list(self, position, element):
        if position > self.size:
            print("Index out of bound")
        elif position == self.size:
            self.add_tail(element)
        elif position == 0:
```

```
        counter = 0
        if index > self.size - 1:
            print("Index out of bound")
            return None
        while counter < index:
            element_node = element_node.next
            counter += 1
        return element_node

    def add_between_list(self, position, element):
        if position > self.size:
            print("Index out of bound")
        elif position == self.size:
            self.add_tail(element)
        elif position == 0:
            self.add_head(element)
        else:
            prev_node = self.get_node_at(position - 1)
            current_node = self.get_node_at(position)
            prev_node.next = element
            element.next = current_node
            self.size -= 1

    def merge(self, linkedlist_value):
        if self.size > 0:
            last_node = self.get_node_at(self.size - 1)
            last_node.next = linkedlist_value.head
            self.size = self.size + linkedlist_value.size
```

Syit_380_umang

Output: