In [1]:

```
class Node():
    def __init__(self, val):
        self.value = val
        self.next = None
        self.prev = None

class DoublyLinkedList():
    def __init__(self):
        self.head = None
        self.tail = None

    def add_node(self, val):
        new_node = Node(val)
        if self.head == None:
            self.head = new_node
            self.tail = new_node
        else:
            current_node = self.head
            while current_node.next != None:
                current_node = current_node.next
            current_node.next = new_node
            new_node.prev = current_node
            self.tail = new_node

    def traverse_forward(self):
        if self.head != None:
            current_node = self.head
            while current_node.next != None:
                print ("Traverse forward current", current_node.value)
                current_node = current_node.next
                print ("Traverse forward current", current_node.value)
        else:
            print ("No nodes")
            return False

    def traverse_back(self):
        if self.tail != None:
            current_node = self.tail
            while current_node.prev != None:
                print ("Traverse backwards current", current_node.value)
                current_node = current_node.prev
            print ("Traverse backwards current", current_node.value)
        else:
            print ("No nodes")
            return False

    def remove_node_from_end(self):
        if self.head != None:
            current_node = self.head
            while current_node.next.next != None:
                current_node = current_node.next
            current_node.next.prev = None
            current_node.next = None
```

```
            self.tail = current_node

    def remove_node(self, val):
        if self.head != None:
            current_node = self.head
            if self.head.value == val:
                self.head.next.prev = None
                self.head = self.head.next
            elif self.tail.value == val:
                self.tail.prev.next = None
                self.tail = self.tail.prev
            else:
                while current_node.next.value != val:
                    current_node = current_node.next
                    if current_node.next.value != val:
                        print ("Value not in list")
                        return False
                current_node.next.next.prev = current_node
                current_node.next = current_node.next.next

    def insert_node_after(self, val, insert_val):
        if self.head != None:
            current_node = self.head
            new_node = Node(insert_val)
            if self.head.value == val:
                self.head.next.prev = new_node
                new_node.prev = self.head
                new_node.next = self.head.next
                self.head.next = new_node
            elif self.tail.value == val:
                self.tail.next = new_node
                new_node.prev = self.tail
                self.tail = new_node
            else:
                while current_node.value != val:
                    current_node = current_node.next
                    if current_node.value != val:
                        print ("Value not in list")
                        return False
                new_node.next = current_node.next
                current_node.next = new_node
                new_node.next.prev = new_node
                new_node.prev = current_node

    def print_as_list(self):
            value_list = []
            if self.head != None:
                current_node = self.head
                while current_node.next != None:
                    value_list.append(current_node.value)
                    current_node = current_node.next
                value_list.append(current_node.value)
                print (value_list)
            else:
```

```
                print ("No nodes")
                return False

dll = DoublyLinkedList()
dll.add_node(10)
dll.add_node(20)
dll.add_node(30)
dll.add_node(35)
dll.print_as_list()
# dll.remove_node_from_end()
# dll.remove_node(35)
# dll.print_as_list()
# dll.remove_node(10)
dll.print_as_list()
dll.insert_node_after(35, 25)
dll.print_as_list()


```

```
[10, 20, 30, 35]
[10, 20, 30, 35]
[10, 20, 30, 35, 25]
```

In [ ]:

```
1
```