Doubly Linked List – Lab Sheet

```
import java.io.*;

class node {

node prev;

int data;

node next;

node(int value) // A constructor is called here

{

// By default previous pointer is pointed to NULL

prev = null;

// Value is assigned to the data

data = value;

// By default next pointer is pointed to NULL

next = null;

}

}

class DLLMain {

// Declaring an empty doubly linked list

static node head = null;

static node tail = null;

static void insertAtBeginning(int data)

{

node temp = new node(data);

if (head == null) {

head = temp;

tail = temp;

}

else {
```

```
temp.next = head;

head.prev = temp;

head = temp;

}

}

static void insertAtEnd(int data)

{

node temp = new node(data);

if (tail == null) {

head = temp;

tail = temp;

}

else {

tail.next = temp;

temp.prev = tail;

tail = temp;

}

}

static void insertAtPosition(int data, int position)

{

node temp = new node(data);

if (position == 1) {

insertAtBeginning(data);

}

else {

node current = head;

int currPosition = 1;

while (current != null
```

```
&& currPosition < position) {
current = current.next;
currPosition++;
}
if (current == null) {
insertAtEnd(data);
}
else {
temp.next = current;
temp.prev = current.prev;
current.prev.next = temp;
current.prev = temp;
}
}
}
static void deleteAtBeginning()
{
if (head == null) {
return;
}
if (head == tail) {
head = null;
tail = null;
return;
}
node temp = head;
head = head.next;
head.prev = null;
```

```
temp.next = null;
}
static void deleteAtEnd()
{
if (tail == null) {
return;
}
if (head == tail) {
head = null;
tail = null;
return;
}
node temp = tail;
tail = tail.prev;
tail.next = null;
temp.prev = null;
}
static void deleteAtSpecificPosition(int pos)
{
if (head == null) {
return;
}
if (pos == 1) {
deleteAtBeginning();
return;
}
node current = head;
int count = 1;
```

```
while (current != null && count != pos) {
current = current.next;
count++;
}
if (current == null) {
System.out.println("Position wrong");
return;
}
if (current == tail) {
deleteAtEnd();
return;
}
current.prev.next = current.next;
current.next.prev = current.prev;
current.prev = null;
current.next = null;
}
static void display(node head)
{
node temp = head;
while (temp != null) {
System.out.print(temp.data + " --> ");
temp = temp.next;
}
System.out.println("NULL");
}
// Drivers code
public static void main(String[] args)
```

```
{
insertAtEnd(1);
insertAtEnd(2);
insertAtEnd(3);
insertAtEnd(4);
insertAtEnd(5);
System.out.print("After insertion at tail: ");
display(head);
System.out.print("After insertion at head: ");
insertAtBeginning(0);
display(head);
insertAtPosition(6, 2);
System.out.print(
"After insertion at 2nd position: ");
display(head);
deleteAtBeginning();
System.out.print(
"After deletion at the beginning: ");
display(head);
deleteAtEnd();
System.out.print("After deletion at the end: ");
display(head);
deleteAtSpecificPosition(2);
System.out.print(
"After deletion at 2nd position: ");
display(head);
}
}
```