**class Node  
{  
 int data;  
 Node next;  
  
 Node(int data, Node next)  
 {  
 this.data = data;  
 this.next = next;  
 }  
}  
  
class Main  
{ public static void printList(String msg, Node head)  
 {  
 System.*out*.print(msg);  
  
 Node ptr = head;  
 while (ptr != null)  
 {  
 System.*out*.print(ptr.data + " —> ");  
 ptr = ptr.next;  
 }  
  
 System.*out*.println("null");  
 }  
  
 public static int findLength(Node head)  
 {  
 int count = 0;  
 Node ptr = head;  
 while (ptr != null)  
 {  
 count++;  
 ptr = ptr.next;  
 }  
 return count;  
 }public static Node[] frontBackSplit(Node source)  
 {  
 Node frontRef, backRef;  
  
 int len = *findLength*(source);  
 if (len < 2)  
 {  
 frontRef = source;  
 backRef = null;  
 return new Node[] { frontRef, backRef };  
 }  
  
 Node current = source;  
  
 int hopCount = (len - 1) / 2; for (int i = 0; i < hopCount; i++) {  
 current = current.next;  
 }  
frontRef = source;  
 backRef = current.next;  
 current.next = null;  
  
 return new Node[] { frontRef, backRef };  
 }  
  
 public static void main(String[] args)  
 {  
int[] keys = {6, 3, 4, 8, 2, 9};  
  
 Node head = null;  
  
 for (int i = keys.length - 1; i >= 0; i--) {  
 head = new Node(keys[i], head);  
 }  
  
 Node[] nodes = *frontBackSplit*(head);  
 *printList*("Front List: ", nodes[0]);  
 *printList*("Back List: ", nodes[1])**

**}**

**2:**

***/\*\*  
 \* Created by op on 3/25/2021.  
 \*/*public class CloneDoubly {  
 public static void main(String[] args) {  
  
 public static void main(String[] args) {  
  
 *// create a LinkedList* DoublyLinkedList <Integer>list=new DoublyLinkedList();  
  
 *// add some elements* list.addFirst(1);  
 list.addFirst(2);  
 DoublyLinkedList <Integer>list2=new DoublyLinkedList<>()  
 list2.addFirst(4);  
 list2.addFirst(6);  
 *// print the list* System.*out*.println("LinkedList:" + list);  
  
 *// clone list1* list2 = (LinkedList) list.clone();  
  
 System.*out*.println("LinkedList 2:" + list2);  
 }  
  
 }  
}**