**Solve (Set#A)**

| **Python** | **Java** |
| --- | --- |
| def reverseSegment(start, stop):  prev\_node = stop.next  curr\_node = start  while curr\_node != prev\_node:  next\_node = curr\_node.next  curr\_node.next = curr\_node.prev  curr\_node.prev = next\_node  curr\_node = next\_node  def reverseDLLBetweenTwoPositions(head, m, n):  start = nodeAt(head, m)  stop = nodeAt(head, n)    before\_start = start.prev  after\_stop = stop.next  reverseSegment(start, stop)  before\_start.next = stop  stop.prev = before\_start  start.next = after\_stop  after\_stop.prev = start  return head | public static void reverseSegment(DoublyNode start, DoublyNode stop) {  DoublyNode prevNode = stop.next;  DoublyNode currNode = start;  while (currNode != prevNode) {  DoublyNode nextNode = currNode.next;  currNode.next = currNode.prev;  currNode.prev = nextNode;  currNode = nextNode;  }  }  public static void reverseDLLBetweenTwoPositions(DoublyNode head, int m, int n) {  DoublyNode start = nodeAt(head, m);  DoublyNode stop = nodeAt(head, n);  DoublyNode beforeStart = start.prev;  DoublyNode afterStop = stop.next;  reverseSegment(start, stop);  beforeStart.next = stop;  stop.prev = beforeStart;  start.next = afterStop;  afterStop.prev = start;  }  } |

**Rubric (Set#A)**

| **SI** | **Category** | **Marks** |
| --- | --- | --- |
| 1 | Proper Method/Function Declaration | 2 |
| 2 | Correct Utilization of the nodeAt() Method | 1 |
| 3 | Maintaining the Dummy Headed Circular Doubly Linked List Structure | 3 |
| 4 | Properly Iterating Through the Nodes for Reversal | 2 |
| 5 | Reversing Only the Targeted Segment Without Affecting Other Nodes | 3 |
| 6 | Reconnecting the Reversed Segment Correctly | 3 |
| 7 | Proper Return Statement | 1 |
| **Total = 15** | | |

**Solve (Set # B)**

| **Python** | **Java** |
| --- | --- |
| def reverseSegment(start, stop):  prev\_node = stop.next  curr\_node = start  while curr\_node != prev\_node:  next\_node = curr\_node.next  curr\_node.next = curr\_node.prev  curr\_node.prev = next\_node  curr\_node = next\_node  def reverseDLLBetweenTwoValues(head, x, y):  start = getNodeByValue(head, x)  stop = getNodeByValue(head, y)  before\_start = start.prev  after\_stop = stop.next  reverseSegment(start, stop)  before\_start.next = stop  stop.prev = before\_start  start.next = after\_stop  after\_stop.prev = start  return head | public static void reverseSegment(DoublyNode start, DoublyNode stop) {  DoublyNode prevNode = stop.next;  DoublyNode currNode = start;  while (currNode != prevNode) {  DoublyNode nextNode = currNode.next;  currNode.next = currNode.prev;  currNode.prev = nextNode;  currNode = nextNode;  }  }  public static void reverseDLLBetweenTwoValues(DoublyNode head, int x, int y) {  DoublyNode start = getNodeByValue(head, x);  DoublyNode stop = getNodeByValue(head, y);  DoublyNode beforeStart = start.prev;  DoublyNode afterStop = stop.next;  reverseSegment(start, stop);  beforeStart.next = stop;  stop.prev = beforeStart;  start.next = afterStop;  afterStop.prev = start;  } |

**Rubric (Set#B)**

| **SI** | **Category** | **Marks** |
| --- | --- | --- |
| 1 | Proper Method/Function Declaration | 2 |
| 2 | Correct Utilization of the getNodeByValue() Method | 1 |
| 3 | Maintaining the Dummy Headed Circular Doubly Linked List Structure | 3 |
| 4 | Properly Iterating Through the Nodes for Reversal | 2 |
| 5 | Reversing Only the Targeted Segment Without Affecting Other Nodes | 3 |
| 6 | Reconnecting the Reversed Segment Correctly | 3 |
| 7 | Proper Return Statement | 1 |
| **Total = 15** | | |