Solve

## **SET#A**

| **PYTHON** | **JAVA** |
| --- | --- |
| def nodeAt(head, index):  current = head  count = 0  while current:  if count == index:  return current  current = current.next  count += 1  return None  def isPalindrome(head):  length = 0  current = head  while current:  length += 1  current = current.next    for i in range(length // 2):  node1 = nodeAt(head, i)  node2 = nodeAt(head, length - 1 - i)  if node1.value != node2.value:  return False  return True | public static Node nodeAt(ListNode head, int index) {  Node current = head;  int count = 0;  while (current != null) {  if (count == index) {  return current;  }  current = current.next;  count++;  }  return null;  }  public static boolean isPalindrome(Node head) {  int length = 0;  Node current = head;  while (current != null) {  length++;  current = current.next;  }  for (int i = 0; i < length / 2; i++) {  Node node1 = nodeAt(head, i);  Node node2 = nodeAt(head, length - 1 - i);  if (node1.value != node2.value) {  return false;  }  }  return true;  } |

## **RUBRIC**

| **Category** | **Marks** |
| --- | --- |
| Properly method/function declaration | 2 |
| Writing the nodeAt() method/function or doing the similar operation inside of the method | 4 |
| Finding out the length | 2 |
| Properly iterating through the nodes to compare | 2 |
| Comparing the values | 2 |
| Breaking out if mismatch or using flag concept | 2 |
| Proper Return Statement | 1 |

## 

## **SET#B**

| **PYTHON** | **JAVA** |
| --- | --- |
| def nodeAt(head, index):  current = head  count = 0  while current:  if count == index:  return current  current = current.next  count += 1  return None  def revMatch(head1, head2):  length = 0  current = head1  while current:  length += 1  current = current.next  for i in range(length):  node1 = nodeAt(head1, i)  node2 = nodeAt(head2, length - 1 - i)  if node1.value != node2.value:  return False  return True | public static Node nodeAt(ListNode head, int index) {  Node current = head;  int count = 0;  while (current != null) {  if (count == index) {  return current;  }  current = current.next;  count++;  }  return null;  }  public static boolean revMatch(Node head1, Node head2) {  int length = 0;  Node current = head1;  while (current != null) {  length++;  current = current.next;  }  for (int i = 0; i < length; i++) {  Node node1 = nodeAt(head1, i);  Node node2 = nodeAt(head2, length - 1 - i);  if (node1.value != node2.value) {  return false;  }  }  return true;  } |

## **RUBRIC**

| **Category** | **Marks** |
| --- | --- |
| Properly method/function declaration | 2 |
| Writing the nodeAt() method/function or doing the similar operation inside of the method | 4 |
| Finding out the length | 2 |
| Properly iterating through the both the nodes to compare | 2 |
| Comparing the values | 2 |
| Breaking out if mismatch or using flag concept | 2 |
| Proper Return Statement | 1 |

## 