**SetA Solution:**

=======♨️=======  
public boolean isPalindrome(Node head) {

Node left = head.next, right = head.prev;

while (left != right && left.prev != right) {

if (left.elem != right.elem) return false;

left = left.next;

right = right.prev;

}

return true;

}

=====🐍=====

def isPalindrome(head):

left, right = head.next, head.prev

while left != right and left.prev != right:

if left.elem != right.elem:

return False

left = left.next

right = right.prev

return True

| Task | Marks |
| --- | --- |
| Initializes left and right properly Node left = head.next;  Node right = head.prev; | 3 |
| While loop condition: → Odd length list: left != right → Even length list: left.prev != right | 4 |
| Equality checking and return correctly  left.elem != right.elem | 3 |
| Updates left and right properly left = left.next  right = right.prev | 5 |

**SetB Solution:**

=====♨️======

public Node[] findPair(Node head, int target) {

Node left = head.next, right = head.prev;

while (left != right && left.prev != right) {

int sum = left.elem + right.elem;

if (sum == target) return new Node[]{left, right};

else if (sum < target) left = left.next;

else right = right.prev;

}

return null;

}

=====🐍=====

def findPair(head, target):

left, right = head.next, head.prev

while left != right and left.prev != right:

sum\_value = left.elem + right.elem

if sum\_value == target:

return (left, right)

elif sum\_value < target:

left = left.next

else:

right = right.prev

return None

| Task | Marks |
| --- | --- |
| Initialize pointers correctly  left = head.next, right = head.prev | 3 |
| Correct while loop condition  left != right && left.prev != right | 4 |
| Check sum and return first valid pair | 3 |
| Update pointers correctly  left = left.next  right = right.prev | 5 |