**Question: Check if a Doubly Linked List is a Palindrome**

You are given a dummy-headed doubly circular linked list. Implement a function to check if the linked list is a palindrome. A list is a palindrome if the elements read the same forward and backward.

* The function should return true if it is a palindrome, otherwise false.
* You can access elem, next, and prev of a node directly. No need to write the node class code.
* No need to write the driver class code.
* You cannot modify the list while checking.

| **Sample Input** | **Sample Output** |
| --- | --- |
| **DH ↔ 1 ↔ 2 ↔ 3 ↔ 2 ↔ 1** | **True** |
| **DH ↔ 1** | **True** |
| **DH** | **True** |
| **DH ↔ 4 ↔ 5 ↔ 6 ↔ 7 ↔ 8** | **False** |

You only need to complete the following functions. You don’t have to write anything else.

| **Java** | **Python** |
| --- | --- |
| public boolean isPalindrome(Node head) {  // TODO: Implement your solution  } | def isPalindrome(head):  # TODO: Implement your solution |

**Question: Find a Pair with a Target Sum in a Sorted Doubly Linked List**

You are given a sorted dummy-headed doubly circular linked list and a target sum, X. Implement a function that returns any first pair of nodes (a, b) such that a.elem + b.elem == X.

* If multiple pairs exist, return the first found pair. If no such pair exists, return None.
* The function should traverse from both forward and backward.
* You can access elem, next, and prev of a node directly. No need to write the node class code.
* No need to write the driver class code.

| **Sample Input** | **Sample Output** |
| --- | --- |
| **DH ↔ 1 ↔ 2 ↔ 3 ↔ 4 ↔ 6 ↔ 8 ↔ 9**  **Target: 10** | **(1, 9)** |
| **DH ↔ 1 ↔ 3 ↔ 5 ↔ 7 ↔ 9 ↔ 10**  **Target: 12** | **(3, 9)** |
| **DH ↔ 1 ↔ 2 ↔ 4 ↔ 5 ↔ 7**  **Target: 20** | **None** |
| **DH**  **Target: 20** | **None** |

**Hint:** Use two pointers. One starting from the smallest element (head.next) and another from the largest (head.prev). Move the front pointer forward if the sum is too small and the back pointer backwards if the sum is too large. Stop when a pair is found or pointers cross.

You only need to complete the following functions. You don’t have to write anything else.

| **Java** | **Python** |
| --- | --- |
| public Node[] findPair(Node head, int target) {  // TODO: Implement your solution  } | def findPair(head, target):  # TODO: Implement your solution |