A teacher gives a child a set of toy boxes arrangedin a circle, each placed in a specific position. As a challenge, the teacher asks the child to reverse the order of boxes between two specific positions **m** and **n**, while keeping the rest of the arrangement unchanged. Now, your task is to help the child fix the arrangement. The boxes are represented as a **Dummy Headed Circular Doubly Linked List**, where each node corresponds to a box.

You are given the **head** of the list and two integers **m** and **n** (1 ≤ m < n ≤ length of the list). **Reverse** the nodes from position m to n without affecting the rest of the list, and return the **modified list's head**. Consider that the **DoublyNode** class, **createDLL()**, **printDLL(), nodeAt()** methods/functions are already provided.

| **Sample Function Call** | **Sample Returned Result** |
| --- | --- |
| **head =>** 10 ⇌ 40 ⇌ 30 ⇌ 20 ⇌ 50,  m = 2, n = 4  **reverseDLLBetweenTwoPositions(head, m, n)** | **Head of the modified linked list =>**  10 ⇌ 20 ⇌ 30 ⇌ 40 ⇌ 50 |

***Note:*** *Solve this task in* ***constant space complexity****, meaning you must modify the linked list* ***in-place*** *without creating a new one. m and n will never have the same value, and the linked list will always contain at least two nodes.*

**\*nodeAt(head, idx):** returns the node at the given index in a Dummy Headed Circular Doubly Linked List or None if out of bounds.

A teacher gives a child a set of toy boxes arranged in a circle, each labeled with a **unique number**. As a challenge, the teacher asks the child to reverse the order of boxes between two specific boxes with given values **x** and **y**, while keeping the rest of the arrangement unchanged. Now, your task is to help the child organize the boxes. The toy boxes are represented as a **Dummy Headed Circular Doubly Linked List**, where each node corresponds to a box.

You are given the **head** of the list and two distinct values **x** and **y** (where x appears before y in the list). **Reverse** the nodes from value x to value y without affecting the rest of the list, and return the **modified list's head**. Consider that the **DoublyNode** class, **createDLL()**, **printDLL(), getNodeByValue()** methods/functions are already provided.

| **Sample Function Call** | **Sample Returned Result** |
| --- | --- |
| **head =>** 10 ⇌ 20 ⇌ 30 ⇌ 90 ⇌ 40 ⇌ 50,  x = 20, y = 40  **reverseDLLBetweenTwoValues(head, x, y)** | **Head of the modified linked list =>**  10 ⇌ 40 ⇌ 90 ⇌ 30 ⇌ 20 ⇌ 50 |

***Note:*** *Solve this task in* ***constant space complexity****, meaning you must modify the linked list* ***in-place*** *without creating a new one. x will always appear before y in the list, and the linked list will always contain at least two nodes.*

**\*getNodeByValue(head, value):** returns the node containing the given value in a Dummy Headed Circular Doubly Linked List, or None if the value is not found.