Write a function called **isPalindrome(head)**, that takes the head of a linked list in its parameter. The function does a very simple task. The function will return True if the linked list is actually a Palindrome otherwise it’ll return False.

| **Sample Function Call** | **Sample Returned Result** |
| --- | --- |
| **head =>** 10 -> 43 -> 54 -> 43 -> 10  **isPalindrome( head )** | **True** |
| **head =>** 1 -> 41 -> 4 -> 3 -> 10  **isPalindrome( head )** | **False** |

**Hint:** You may want to **write the nodeAt()** method/function to use it in this task.

Write a function called **revMatch(head1, head2)**, that takes two heads of two different linked lists in its parameter. The function checks whether the first linked list is matching with the second linked list in reverse order. If they match up then the function will return True otherwise False. You can assume that the length of these two linked lists are always equal.

| **Sample Function Call** | **Sample Returned Result** |
| --- | --- |
| **head1 =>** 1 -> 3 -> 5 -> 7 -> 4  **head2 =>** 4 -> 7 -> 5 -> 3 -> 1  **revMatch( head1, head2 )** | **True** |
| **head1 =>** 1 -> 9 -> 2 -> 7 -> 4  **head2 =>** 4 -> 7 -> 5 -> 3 -> 1  **revMatch( head1, head2 )** | **False** |

**Hint:** You may want to **write the nodeAt()** method/function to use in this task.