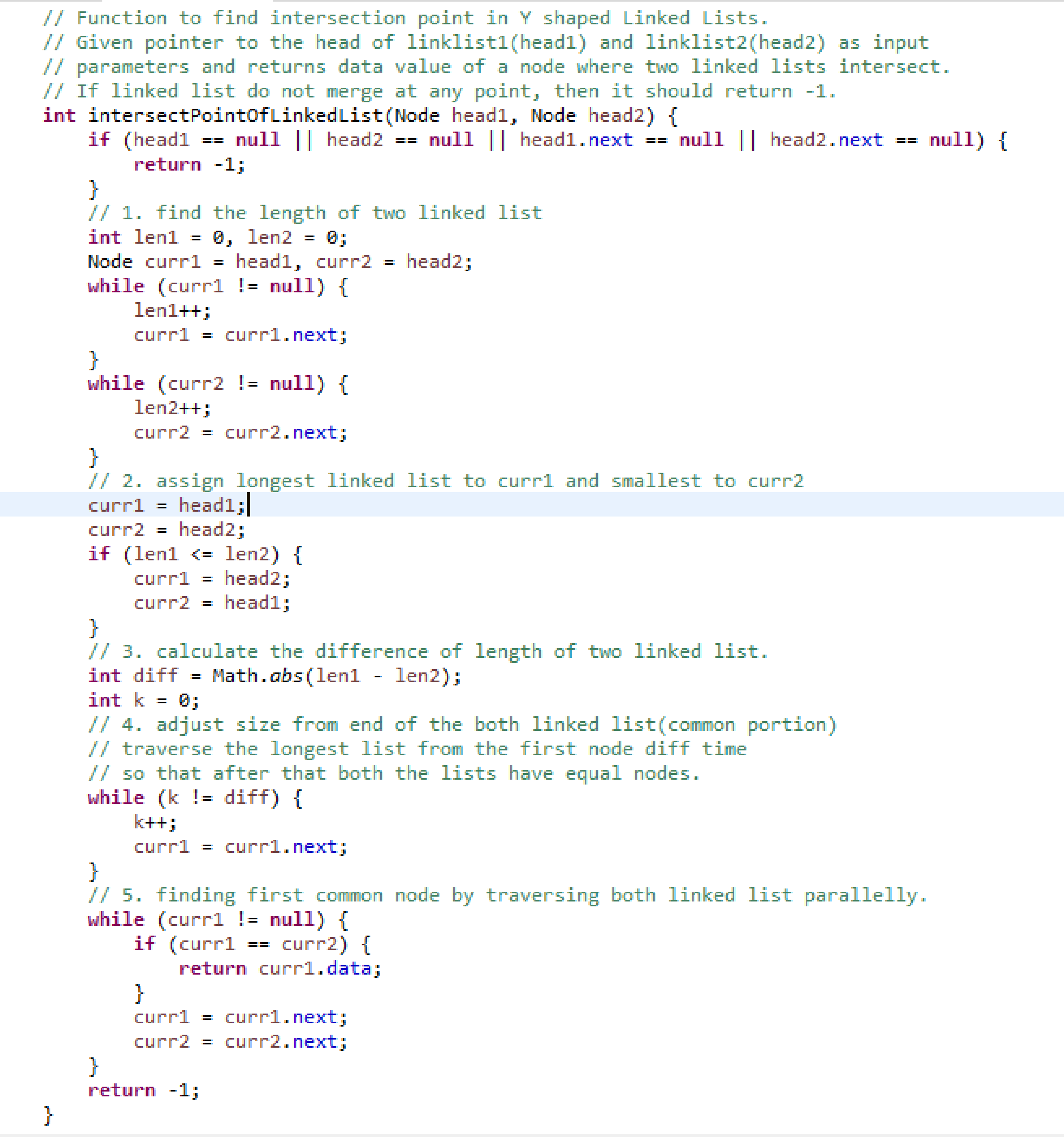

```
// Function to find intersection point in Y shaped Linked Lists.
// Given pointer to the head of linklist1(head1) and linklist2(head2) as input
// parameters and returns data value of a node where two linked lists intersect.
// If linked list do not merge at any point, then it should return -1.
int intersectPointOfLinkedList(Node head1, Node head2) {
    if (head1 == null || head2 == null || head1.next == null || head2.next == null) {
        return -1;
    }
    // 1. find the length of two linked list
    int len1 = 0, len2 = 0;
    Node curr1 = head1, curr2 = head2;
    while (curr1 != null) {
        len1++;
        curr1 = curr1.next;
    }
    while (curr2 != null) {
        len2++;
        curr2 = curr2.next;
    }
    // 2. assign longest linked list to curr1 and smallest to curr2
    curr1 = head1;
    curr2 = head2;
    if (len1 <= len2) {
        curr1 = head2;
        curr2 = head1;
    }
    // 3. calculate the difference of length of two linked list.
    int diff = Math.abs(len1 - len2);
    int k = 0;
    // 4. adjust size from end of the both linked list(common portion)
    // traverse the longest list from the first node diff time
    // so that after that both the lists have equal nodes.
    while (k != diff) {
        k++;
        curr1 = curr1.next;
    }
    // 5. finding first common node by traversing both linked list parallelly.
    while (curr1 != null) {
        if (curr1 == curr2) {
            return curr1.data;
        }
        curr1 = curr1.next;
        curr2 = curr2.next;
    }
    return -1;
}
```