**Name-Suraj Paswan**

**Enrol.no-2002106**

**Div-B**

**Q10- Write a C program to merge two sorted lists using doubly linked list**.  
  
// C# implementation for Sorted merge of two  
// sorted doubly circular linked list using System;  
  
class GFG  
{  
  
public class Node  
{  
public int data;  
public Node next, prev;  
};  
  
// A utility function to insert a new node at the  
// beginning of doubly circular linked list  
static Node insert(Node head\_ref, int data)  
{  
// allocate space  
Node new\_node = new Node();  
  
// put in the data  
[new\_node.data](http://new_node.data/) = data;  
  
// if list is empty  
if (head\_ref == null)  
{  
[new\_node.next](http://new_node.next/) = new\_node;  
[new\_node.prev](http://new_node.prev/) = new\_node;  
}  
  
else  
{  
  
// pointer points to last Node  
Node last = (head\_ref).prev;  
  
// setting up previous and next of new node  
[new\_node.next](http://new_node.next/) = head\_ref;  
[new\_node.prev](http://new_node.prev/) = last;  
  
// update next and previous pointers of head\_ref  
// and last.  
[last.next](http://last.next/) = (head\_ref).prev = new\_node;  
}  
  
// update head\_ref pointer  
head\_ref = new\_node;  
return head\_ref;  
}  
  
// function for Sorted merge of two  
// sorted doubly linked list  
static Node merge(Node first, Node second)  
{  
// If first list is empty  
if (first == null)  
return second;  
  
// If second list is empty  
if (second == null)  
return first;  
  
// Pick the smaller value and adjust  
// the links  
if ([first.data](http://first.data/) < [second.data](http://second.data/))  
{  
[first.next](http://first.next/) = merge([first.next](http://first.next/), second);  
[first.next.prev](http://first.next.prev/) = first;  
[first.prev](http://first.prev/) = null;  
return first;  
}  
else  
{  
[second.next](http://second.next/) = merge(first, [second.next](http://second.next/));  
[second.next.prev](http://second.next.prev/) = second;  
[second.prev](http://second.prev/) = null;  
return second;  
}  
}  
  
// function for Sorted merge of two sorted  
// doubly circular linked list  
static Node mergeUtil(Node head1, Node head2)  
{  
// if 1st list is empty  
if (head1 == null)  
return head2;  
  
// if 2nd list is empty  
if (head2 == null)  
return head1;  
  
// get pointer to the node which will be the  
// last node of the final list  
Node last\_node;  
if ([head1.prev.data](http://head1.prev.data/) < [head2.prev.data](http://head2.prev.data/))  
last\_node = [head2.prev](http://head2.prev/);  
else  
last\_node = [head1.prev](http://head1.prev/);  
  
// store null to the 'next' link of the last nodes  
// of the two lists  
[head1.prev.next](http://head1.prev.next/) = [head2.prev.next](http://head2.prev.next/) = null;  
  
// sorted merge of head1 and head2  
Node finalHead = merge(head1, head2);  
  
// 'prev' of 1st node pointing the last node  
// 'next' of last node pointing to 1st node  
[finalHead.prev](http://finalhead.prev/) = last\_node;  
[last\_node.next](http://last_node.next/) = finalHead;  
  
return finalHead;  
}  
  
// function to print the list  
static void printList(Node head)  
{  
Node temp = head;  
  
while ([temp.next](http://temp.next/) != head)  
{  
[Console.Write](http://console.write/)([temp.data](http://temp.data/) + " ");  
temp = [temp.next](http://temp.next/);  
}  
[Console.Write](http://console.write/)([temp.data](http://temp.data/) + " ");  
}  
  
// Driver code  
public static void Main()  
{  
Node head1 = null, head2 = null;  
  
// list 1:  
head1 = insert(head1, 8);  
head1 = insert(head1, 5);  
head1 = insert(head1, 3);  
head1 = insert(head1, 1);  
  
// list 2:  
head2 = insert(head2, 11);  
head2 = insert(head2, 9);  
head2 = insert(head2, 7);  
head2 = insert(head2, 2);  
  
Node newHead = mergeUtil(head1, head2);  
  
[Console.Write](http://console.write/)( "Final Sorted List: ");  
printList(newHead);  
}  
}  
  
**Output:**  
  
Final Sorted List: [1 2 3 5 7 8 9 11](tel:1%202%203%205%207%208%209%2011)