```
//HW1: due Saturday September 16,  at 11:59pm
/*
Implement member functions and  Merge.  Your implementation has to be in-place.
You are required to implement the Merge algorithm to be described in class.
You are not allowed to use any external structures (such as array)
You are not allowed to use any helper function
You are not allowed to create new nodes or any extra structures (such as array, list, etc.).
*/
#include <iostream>
using namespace std;

class Dnode {
public:
        int value;
        Dnode* next;
        Dnode* previous;
        Dnode(int i) { value = i; next = previous = nullptr; }
        Dnode() { next = previous = nullptr; }
};

class DLL {//Doubly Linked List
public:
        Dnode* head;
        Dnode* tail;
        DLL(int n, int m);//Constructor; Construct an n-node DLL with random
        //values in 0 ... m-1
        DLL() { head = tail = nullptr; }
        void PrintF();//Print forward from head
        void PrintB();//Print backward from tail


        void Sort(Dnode * p1, Dnode *p2);//Sort the portion of a DLL from
        //the Dnode pointed by p1 until the Dnode pointed by p2


        void Merge(Dnode* p1, Dnode* p2, Dnode* p3, Dnode* p4);
        //Merge the two portions of the Linked List, as described in clas and the video
recording for merge algoirthm.
        //Note that each of the two portions is already sorted before the merge.

};


DLL::DLL(int n, int m) {
        head = tail = nullptr;
        for (int i = 0; i < n; ++i) {
                Dnode* p1{ new Dnode{rand() % m} };
                if (!head) {//empty
                        head = tail = p1;
                }
                else {
                        tail->next = p1;
                        p1->previous = tail;
                        tail = p1;
                }
        }
}

void DLL::PrintF() {
        cout << endl;
        Dnode* p1{ head };
        while (p1) {
                cout << p1->value << " ";
```

```
                p1 = p1->next;
        }
}

void DLL::PrintB() {
        cout << endl;
        Dnode* p1{ tail };
        while (p1) {
                cout << p1->value << " ";
                p1 = p1->previous;
        }
}



int main() { //During grading, TA might use differnt examples to test your code.
        DLL  L1{ 30, 20 };
        L1.PrintF();
        L1.PrintB();

        L1.Sort(L1.head, L1.tail );
        L1.PrintF();
        L1.PrintB();
        cout << endl;
        DLL L2{ 29, 15 };
        L2.Sort(L2.head, L2.tail);
        L1.tail->next = L2.head;
        L2.head->previous = L1.tail;
        L1.tail = L2.tail;

        L1.Merge(L1.head->next, L2.head->previous->previous, L2.head, L1.tail->previous);

        L1.PrintF();
        L1.PrintB();


        return 0;
}
```