#include <iostream>

using namespace std;

class Node {

private:

int Number;

Node\* Next;

Node\* Prior;

public:

Node(){

this->Number = 0;

this->Next = nullptr;

this->Prior = nullptr;

}

Node(const int& n);

const int& GetNumber() const {

return Number;

}

const Node\* GetNext()const {

return Next;

}

const Node\* GetPrior()const {

return Prior;

}

friend class LinkedList;

};

Node::Node(const int& n) {

this->Number = n;

this->Next = nullptr;

this->Prior = nullptr;

}

class LinkedList {

private:

Node\* Head;

Node\* Tail;

public:

LinkedList() {

Head = nullptr;

Tail = nullptr;

}

void Clear();//清除堆空间

~LinkedList() { Clear(); }//析构函数

const Node\* GetHead()const { return Head; }

Node\* Add(const int& Data);//插入在尾部

//Node\* InsertFrist(const int& Data);//插入在第一个

const int GetCount();//计总数

const int GetThisCount(Node\* This);

int CheckMaxPlace();

void Combine(LinkedList& L1, LinkedList& L2);

//const Node\* operator[] (int index)const;

//Node\* operator[](int index);

//Node\* Insert(const int& Data,int index);//插入在index处

};

void LinkedList::Clear() {

while (Head) {

Node\* newHead = Head->Next;

delete Head;

Head = newHead;

}

}

Node\* LinkedList::Add(const int& Data) {

Node\* newNode = new Node(Data);

if (!Tail)

Head = Tail = newNode;

else

{

Tail->Next = newNode;

Tail = newNode;

}

return newNode;

}

const int LinkedList::GetCount() {

int i = 0;

for (Node\* Count = Head; Count; Count = Count->Next)i++;

return i;

}

const int LinkedList::GetThisCount(Node\* This) {

int i = 1;

for (Node\* Count = Head; (Count)&&(Count != This); Count = Count->Next)i++;

return i;

}

int LinkedList::CheckMaxPlace() {

int n = Head->Number,i=1;

Node\* This=nullptr;

for (Node\* Count = Head->Next; Count; Count = Count->Next) {

if (n < Count->Number) {

n = Count->Number;

This = Count;

}

}

i = GetThisCount(This);

return i;

}

void LinkedList::Combine(LinkedList& L1, LinkedList& L2) {//将L1和L2合并进L1，并把L2归空

if (L1.Head && L2.Head) {

Node\* p=L1.Head, \*q=L2.Head ;

if (p->Number > q->Number) {

L1.Head = q;

q->Next ->Prior =q;

q = q->Next;

q->Prior ->Next =p;

}

while (p->Next || q->Next ) {

while (p->Next && q->Next) {

if (p->Number <= q->Number) {

p->Next->Prior = p;

p = p->Next;

}

else {

p->Prior->Next = q;

q->Next->Prior = q;

q = q->Next;

q->Prior->Next = p;

}

}

while (!p->Next && q->Next ) {//p是最后一位，q不是

if (p->Number <= q->Number) {

p->Next = q;

L1.Tail = q;

break;

}

else {

p->Prior->Next = q;

q->Next->Prior = q;

q = q->Next;

q->Prior->Next = p;

}

}

while(p->Next && !q->Next){//q是最后一位，p不是

if (p->Number <= q->Number) {

p->Next->Prior = p;

p = p->Next;

}

else {

p->Prior->Next = q;

q->Next = q;

break;

}

}

break;

}

if (p->Number <= q->Number) {//都是最后一位

p->Next = q;

L1.Tail = q;

}

else {

p->Prior->Next = q;

q->Next = q;

}

}

else {

if (!L1.Head)L1.Head = L2.Head;

}

L2.Head = nullptr;

}

ostream& operator << (ostream& o, const LinkedList& list)

{

for (const Node\* p = list.GetHead();

p;

p = p->GetNext())

{

o << "[" << p->GetNumber() << "]";

if (p->GetNext())

o << " -> ";

}

return o;

}

int main()

{

LinkedList Mylist1;

Mylist1.Add(1);

Mylist1.Add(2);

Mylist1.Add(7);

Mylist1.Add(16);

Mylist1.Add(22);

LinkedList Mylist2;

Mylist2.Add(0);

Mylist2.Add(4);

Mylist2.Add(12);

Mylist2.Add(18);

Mylist2.Add(33);

cout << "Mylist1 is" << endl;

cout << Mylist1<< endl;

cout << "Mylist2 is" << endl;

cout << Mylist2 << endl;

Mylist1.Combine(Mylist1, Mylist2);

cout<< "Mylist is" << endl;

cout << Mylist1 << endl;

//cout << "The length is " << Mylist.GetCount()<< endl;

//cout << "The maxnumber is at " << Mylist.CheckMaxPlace() << endl;

}

