#include<iostream>

using namespace std;

template<class T>

struct Node

{

T data;

struct Node<T>\* next;

};

template <class T>

class LinkList

{

public:

LinkList() { front = new Node<T>; front->next = NULL; }//无参构造函数

LinkList(T a[], int n); //有参构造函数，使用含有n个元素的数组a初始化单链表

~LinkList(); //析构

void PrintList(); //打印

int GetLength() { return length; } //获取长度

void Merge(LinkList <T> &L); //合并

Node<T> \*front; //头指针

private:

int length;

};

template<class T>

LinkList<T>::LinkList(T a[], int n)

{

length = n;

front = new Node<T>;

Node<T>\*r = front;

for (int i = 0; i < n; i++)

{

r->next = new Node<T>;

r = r->next;

r->data = a[i];

}

}

template<class T>

LinkList<T>::~LinkList()

{

Node<T>\*p = front;

while (p)

{

front = p;

p = p->next;

delete front;

}

}

template<class T>

void LinkList<T>::PrintList()

{

Node<T>\*r = front;

while (r->next != 0)

{

cout << r->data << " ";

r = r->next;

}

}

template<class T>

void LinkList<T>::Merge(LinkList<T>&L)

{

int m, n, sumlength;

m = GetLength();

n = L.GetLength();

sumlength = m + n;

int \* p = new int[sumlength];

Node<T>\*s = front->next;

for (int i = 0; i < m; i++)

{

p[i] = s->data;

s = s->next;

}

Node<T>\*s1 = L.front->next;

for (int i = 0; i < n; i++)

{

p[i + m] = s1->data;

s = s->next;

}

for (int i = 0; i < sumlength; i++)

cout << p[i];

}

int main()

{

LinkList <int>L1, L2;

int m, n;

cin >> m;

int \*p1 = new int[m];

LinkList<int>(p1, m);

cin >> n;

int \*p2 = new int[n];

LinkList<int>(p2, n);

L1.Merge(L2);

/\*L1.PrintList();//打印结果链表\*/

delete[]p1;

delete[]p2;

return 0;

}