// two codes for same

#include<bits/stdc++.h>

using namespace std;

class node{

public:

int data;

node\* next;

};

class linked\_list{

private:

node\* head;

node\* tail;

int size;

public:

linked\_list(){

head = NULL;

tail = NULL;

size = 0;

}

node\* gethead(){

return head;

}

node\* gettail(){

return tail;

}

int getsize(){

return size;

}

void addlast(int value){

node\* temp = new node();

temp->data = value;

temp->next = NULL;

if(head==NULL){

head = temp;

tail = temp;

}

else{

tail->next = temp;

tail = temp;

}

size++;

}

void addfirst(int value){

node \*temp = new node();

temp->data = value;

if(head==NULL){

temp->next = NULL;

head = temp;

tail = temp;

}

else{

temp->next = head;

head = temp;

}

size++;

}

void display(){

node\* temp = head;

while(temp!=0){

cout<<temp->data<<" ";

temp = temp->next;

}

cout<<endl;

}

static int intersection(linked\_list one, linked\_list two){

node\* h1 = one.gethead();

node\* h2 = two.gethead();

int s1 = one.getsize();

int s2 = two.getsize();

int diff = abs(s1-s2);

if(s1>s2){

for(int i=0;i<diff;i++){

h1 = h1->next;

}

}

else{

for(int i=0;i<diff;i++){

h2 = h2->next;

}

}

while(h1->data!=h2->data){

h1 = h1->next;

h2 = h2->next;

}

return h1->data;

}

static int inter(node\* h1, node\* h2){

node\* p1 = h1;

node\* p2 = h2;

while(p1!=NULL && p2!=NULL && p1!=p2){

p1 = p1->next;

p2 = p2->next;

if(p1->data = p2->data){

return p1->data;

}

if(p1==NULL){

p1 = h2;

}

if(p2==NULL){

p2 = h1;

}

}

return p1->data;

}

static int addhelper(node\* h1, node\* h2, int pv1, int pv2, linked\_list res){

if(h1==NULL && h2==NULL){

return 0;

}

int d = 0;

if(pv1>pv2){

int c = addhelper(h1->next,h2,pv1-1,pv2,res);

d = h1->data + c;

}

else if(pv1<pv2){

int c = addhelper(h1,h2->next,pv1,pv2-1,res);

d = h2->data + c;

}

else{

int c = addhelper(h1->next,h2->next,pv1-1,pv2-1,res);

d = c + h1->data + h2->data;

}

int num = d%10;

int carry = d/10;

res.addlast(num);

// res.display();

return carry;

}

static linked\_list add(linked\_list l1, linked\_list l2){

linked\_list res;

int c = addhelper(l1.gethead(),l2.gethead(),l1.getsize(), l2.getsize(),res);

if(c>0){

res.addlast(c);

}

// res.display();

return res;

}

};

int main(){

linked\_list l1;

linked\_list l2;

int n,m;

cin>>n;

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

int num;

cin>>num;

l1.addlast(num);

}

cin>>m;

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

int num;

cin>>num;

l2.addlast(num);

}

// linked\_list res;

// int c = linked\_list::addhelper(l1.gethead(),l2.gethead(),l1.getsize(), l2.getsize(),res);

// if(c>0){

// res.addlast(c);

// }

linked\_list c = linked\_list::add(l1,l2);

c.display();

}

----------------------------------------------------------------------------------------------------------------------------

#include <iostream>

using namespace std;

struct node

{

int data;

node \*next;

};

class linked\_list

{

public:

node \*head,\*tail;

int size;

linked\_list()

{

head = NULL;

tail = NULL;

size = 0;

}

void addfirst(int value){

node\* temp = new node;

temp->data = value;

if(head==NULL){

temp->next = NULL;

tail = temp;

head = temp;

}

else{

temp->next = head;

head = temp;

}

size++;

}

void addlast(int n)

{

node \*tmp = new node;

tmp->data = n;

tmp->next = NULL;

if(head == NULL)

{

head = tmp;

tail = tmp;

}

else

{

tail->next = tmp;

tail = tail->next;

}

size++;

}

node\* gethead()

{

return head;

}

node\* gettail()

{

return tail;

}

int getsize(){

return size;

}

static node\* mid\_node(node\* head, node\* tail){

node\* slow = head;

node\* fast = head;

while(fast!=tail && fast->next!=tail){

slow = slow->next;

fast = fast->next->next;

}

return slow;

}

static void display(node \*head)

{

node\* temp = head;

while(temp!=NULL){

cout<<temp->data<<" ";

temp = temp->next;

}

cout<<endl;

}

static linked\_list mergeTwolist(node \*a,node \*b)

{

linked\_list c;

node\* i = a;

node\* j = b;

while(i!=NULL && j!=NULL){

if(i->data<=j->data){

c.addlast(i->data);

i = i->next;

}

else{

c.addlast(j->data);

j = j->next;

}

}

while(i!=NULL){

c.addlast(i->data);

i = i->next;

}

while(j!=NULL){

c.addlast(j->data);

j = j->next;

}

return c;

}

static linked\_list merge\_sort(node\* head, node\* tail){

if(head==tail){

linked\_list base;

base.addlast(head->data);

// cout<<head->data<<endl;

return base;

}

node\* mid = linked\_list::mid\_node(head,tail);

linked\_list s1 = linked\_list::merge\_sort(head,mid);

linked\_list s2 = linked\_list::merge\_sort(mid->next, tail);

linked\_list res = linked\_list::mergeTwolist(s1.gethead(),s2.gethead());

return res;

}

static int addhelper(node\* one, int pv1, node\* two, int pv2, linked\_list res){

if(one==NULL && two==NULL){

return 0;

}

int d = 0;

if(pv1>pv2){

int oc = addhelper(one->next,pv1-1,two,pv2,res);

d = one->data + oc;

}

else if(pv2>pv1){

int oc = addhelper(one,pv1,two->next,pv2-1,res);

d = one->data + oc;

}

else{

int oc = addhelper(one->next,pv1-1,two->next,pv2-1,res);

d = one->data + two->data + oc;

}

int carry = d/10;

int num = d%10;

res.addfirst(num);

cout<<num<<" "<<carry;

cout<<endl;

return carry;

}

static linked\_list add(linked\_list a, linked\_list b){

node\* h1 = a.gethead();

node\* h2 = b.gethead();

int s1 = a.getsize();

int s2 = b.getsize();

linked\_list res;

// cout<<h1->data<<" "<<h2->data<<endl;

// cout<<s1<<" "<<s2<<endl;

int c = linked\_list::addhelper(h1,s1,h2,s2,res);

if(c>0){

res.addfirst(c);

}

cout<<endl;

linked\_list::display(res.gethead());

return res;

}

};

int main()

{

linked\_list a;

linked\_list b;

int n,m,num;

cin>>n;

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

cin>>num;

a.addlast(num);

}

// linked\_list::display(a.gethead());

cin>>m;

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

cin>>num;

b.addlast(num);

}

int num1,num2;

cin>>num1>>num2;

linked\_list c = linked\_list::add(a,b);

linked\_list::display(c.gethead());

c.addfirst(num1);

c.addlast(num2);

linked\_list::display(c.gethead());

return 0;

}