7-1　两个有序链表序列的合并　(20　分)

已知两个非降序链表序列S1与S2，设计函数构造出S1与S2合并后的新的非降序链表S3。

输入格式:

输入分两行，分别在每行给出由若干个正整数构成的非降序序列，用−1表示序列的结尾（−1不属于这个序列）。数字用空格间隔。

输出格式:

在一行中输出合并后新的非降序链表，数字间用空格分开，结尾不能有多余空格；若新链表为空，输出NULL。

输入样例:

```
1 3 5 -1
2 4 6 8 10 -1
```

输出样例:

```
1 2 3 4 5 6 8 10
```

```
#include<stdio.h>
#include <stdlib.h>

#define TRUE        1
#define FALSE       0
#define OK          1
#define ERROR       0
#define INFEASIBLE -1
#define OVERFLOW   -2

typedef struct LNode
{
    int data;
    struct LNode *next;
} LNode,*LinkList;
void ListPrint_L(LinkList L)
{

    LNode *p=L->next;
    if(!p)
        printf("NULL\n");
    while(p!=NULL)
    {
        if(p->next!=NULL)
            printf("%d ",p->data);
        else
            printf("%d\n",p->data);
        p=p->next;
    }
}
```

```
int main()
{
    LinkList La, Lb, Lc;
    int da;
    LNode *curPtr_a, *rearPtr_a, prePtr_a;
    LNode *curPtr_b, *rearPtr_b, prePtr_b;
    LNode *curPtr_c, *rearPtr_c, prePtr_c;
    La = (LinkList)malloc(sizeof(LinkList));
    Lb = (LinkList)malloc(sizeof(LinkList));
    Lc = (LinkList)malloc(sizeof(LinkList));
    Lc->next = NULL;
    rearPtr_a = La, rearPtr_b = Lb, rearPtr_c = Lc;
    scanf("%d", &da);
    while(da != -1)
    {
        curPtr_a = (LNode *)malloc(sizeof(LNode));
        curPtr_a->data = da;
        rearPtr_a->next = curPtr_a;
        rearPtr_a = curPtr_a;
        scanf(" %d", &da);
    }
    rearPtr_a->next = NULL;
    getchar();
    scanf("%d", &da);
    while(da != -1)
    {
        curPtr_b = (LNode *)malloc(sizeof(LNode));
        curPtr_b->data = da;
        rearPtr_b->next = curPtr_b;
        rearPtr_b = curPtr_b;
        scanf(" %d", &da);
    }
     rearPtr_b->next = NULL;
     curPtr_a = La->next;
     curPtr_b = Lb->next;
     while(curPtr_a && curPtr_b)
     {
         if(curPtr_a->data <= curPtr_b->data)
         {
             rearPtr_c->next = curPtr_a;
             rearPtr_c = rearPtr_c->next;
             curPtr_a = curPtr_a->next;
         }
         else
        {
             rearPtr_c->next = curPtr_b;
             rearPtr_c = rearPtr_c->next;
```

```
            curPtr_b = curPtr_b->next;
        }
    }
    if(curPtr_a)
    {
        rearPtr_c->next = curPtr_a;
    }
    if(curPtr_b)
    {
        rearPtr_c->next = curPtr_b;
    }

    ListPrint_L(Lc);
}
```