**1、重建二叉树**

**Description**

给定一个带#号的前序遍历序列，实现以下操作：

1. 重建该二叉树；
2. 中序遍历该二叉树；
3. 找出该二叉树中节点之和最大的路径；

二叉树节点定义如下：

 struct Node

{  
     int value;   
 Node\* left;  
 Node\* right;

};

**Input**

输入一行带#的整数序列，为某二叉树的前序遍历序列。

**Output**

输出两行，第一行为重建该二叉树后的中序遍历序列，第二行为最大路径的和。

**Sample Input**

8，6，3，#，#，7，11，#，#，#，10，#，2，#，5，#，#，#

**Sample Output**

3 6 11 7 8 2 5 10

32

**2、建立有序链表并合并**

**Description**

把两个无序的序列以链表的形式递增有序的存储，即在链表插入的过程中保持链表节点递增有序，之后合并这两个链表并使新链表中的节点仍然是递增排序的。链表节点定义如下：

 struct ListNode

{  
     int value;   
 ListNode\* next;

};

**Input**

输入两行无序整数序列，分别为两个链表的初始元素。

**Output**

输出合并两个有序链表后的递增排序链表序列。

**Sample Input**

3 7 5 9 1

6 0 8 4 2

**Sample Output**

0 1 2 3 4 5 6 7 8 9

**3.图**

**Description**

图中显示的是Emma的人脉网络图：



1. 用邻接表存储该网络图。这里要求每个顶点对其邻接点的访问顺序按照字符串大小来定， 比如顶点Emma对于它的3个邻接点Joyce, Edith, May的访问顺序应该是Edith, Joyce, May.
2. 指定一个人，输出从他开始的广度优先搜索和深度优先搜索。如果给定的人名不在图中，返回-1.

**Input**

1. 输入顶点数和边数；输入顶点信息；输入边的信息
2. 输入一个人名

**Output**

1. 按照输入顶点的顺序，输出每个顶点的邻接点信息(按照字符串大小顺序输出)
2. 输出从指定人开始的广度优先搜索和深度优先搜索

**Sample Input**

8 9

Emma Joyce Edith May Jessie Hailey Gloria Bella

Emma Joyce

Emma Edith

Emma May

Joyce Jessie

Joyce Hailey

Edith Gloria

Jessie Hailey

Hailey Bella

Hailey Gloria

Emma

David

**Sample Output**

Emma Edith Joyce May

Joyce Emma Hailey Jessie

Edith Emma Gloria

May Emma

Jessie Hailey Joyce

Hailey Bella Gloria Jessie Joyce

Gloria Edith Hailey

Bella Hailey

Emma Edith Joyce May Gloria Hailey Jessie Bella

Emma Edith Gloria Hailey Bella Jessie Joyce May

-1