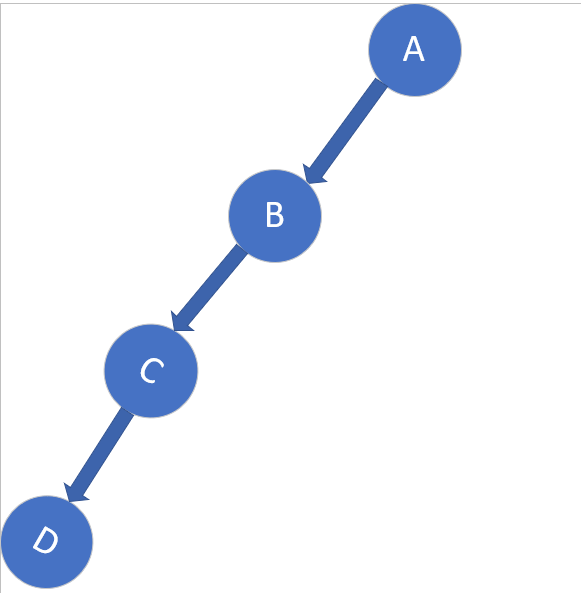
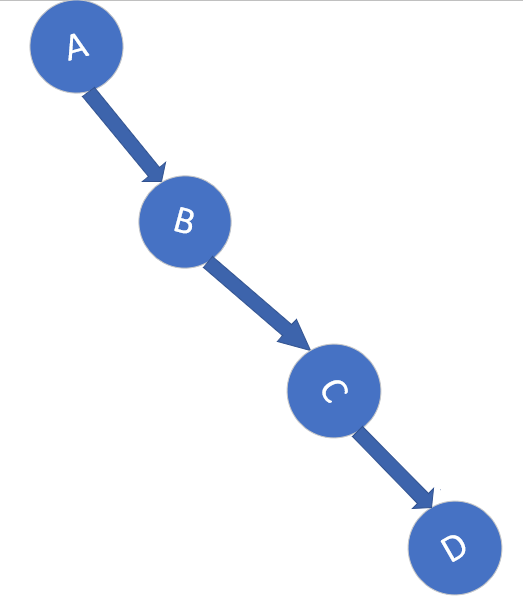
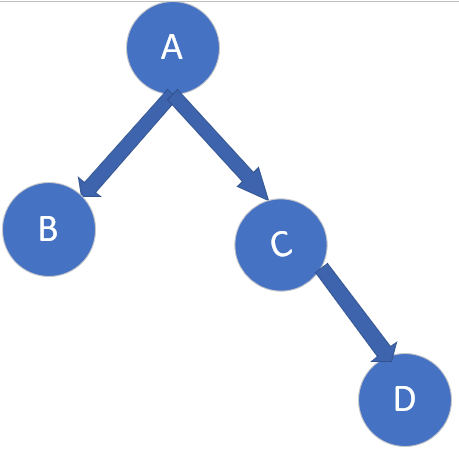
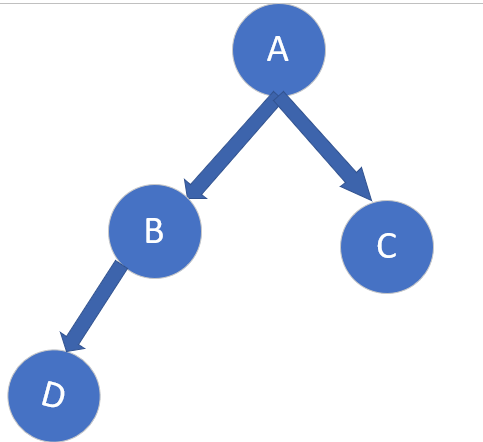
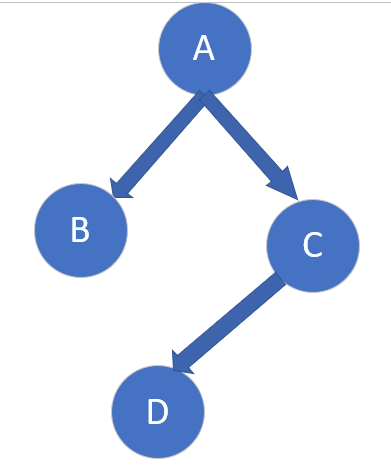
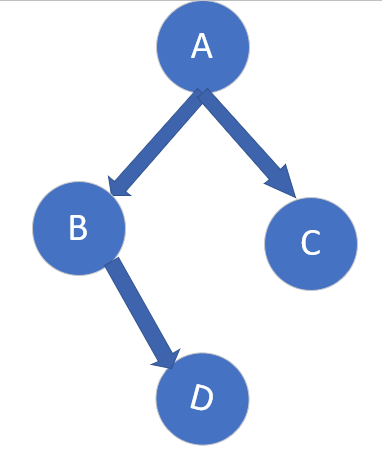
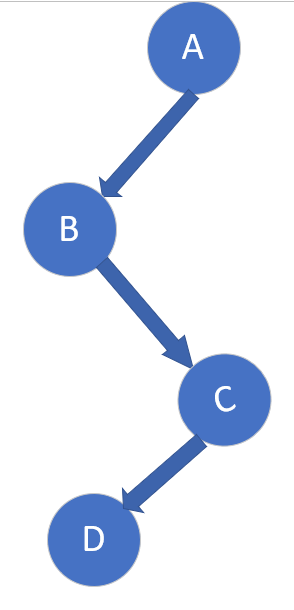
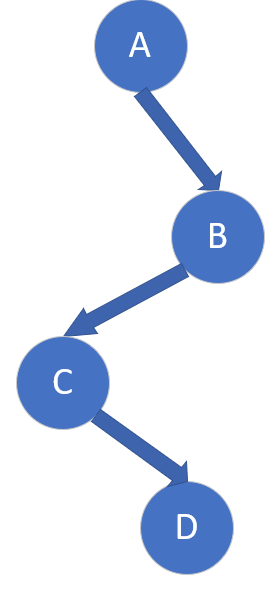
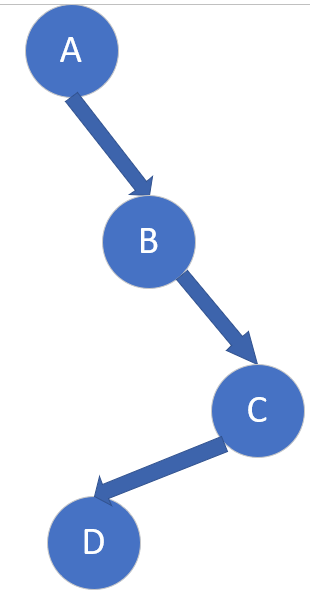
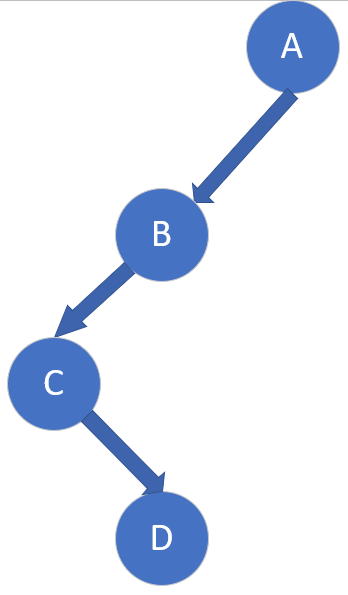
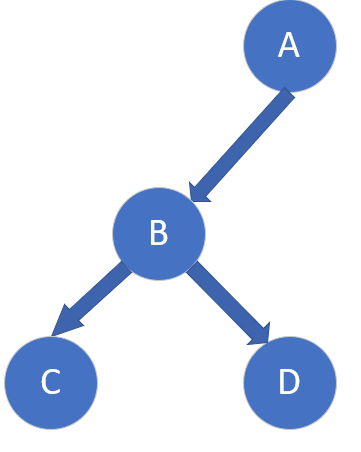
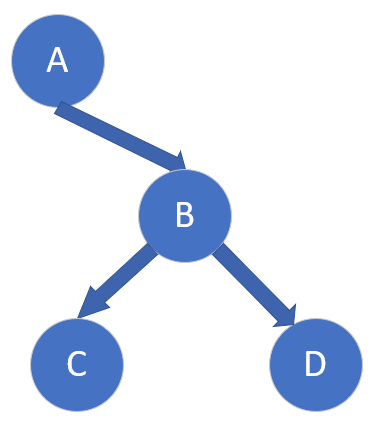
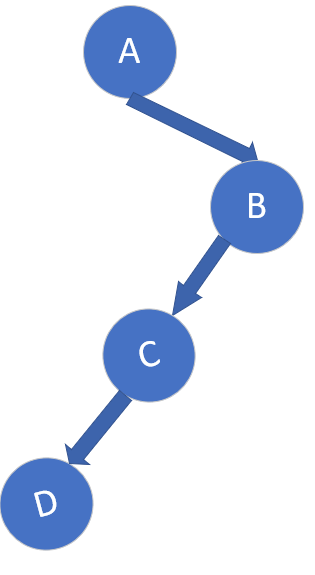
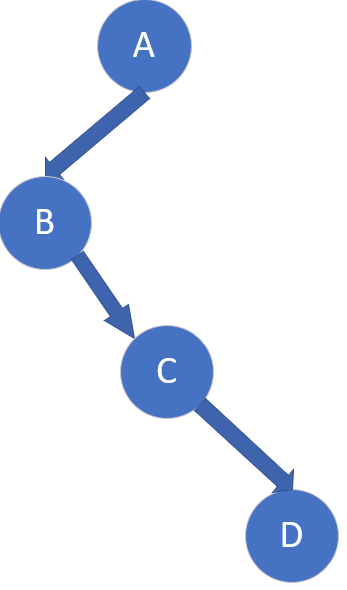
**第五章作业**

第一题：5.1 画出由 4 个结点所构成的所有形态的树（假设是无序树）。

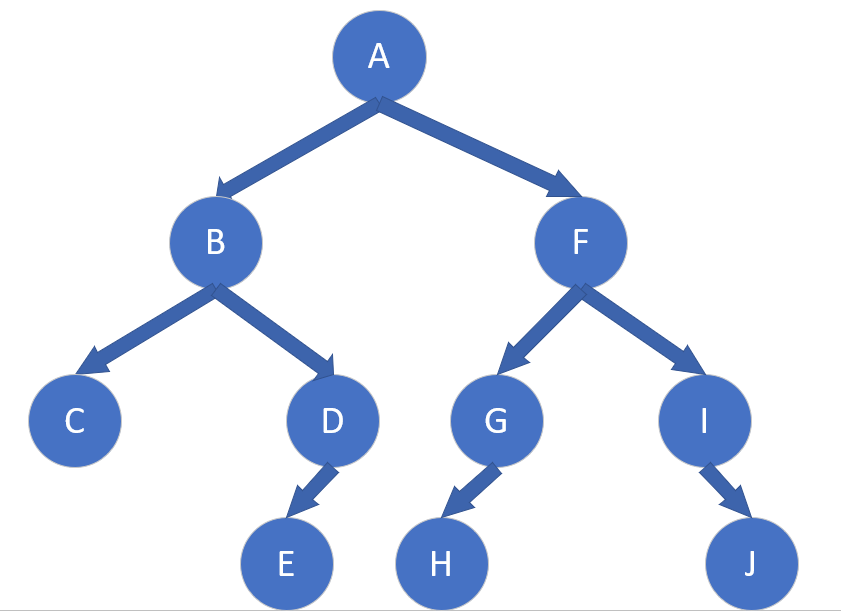
第二题：5.12 已知一棵二叉树的先序、中序和后序序列如下，其中各有一部分未给出其值，请构 造出该二叉树。

先序：A\_CDEF\_H\_J 补全之后 先序：ABCDEFGHIJ

中序：C\_EDA\_GFI\_ 中序：CBEDAHGFIJ

后序：C\_ \_BHGJI\_ \_ 后序：CEDBHGJIFA

由先序确定根节点，中序根据根结点A,将序列分为左右子树，再对左右子树构造分别进行分析调整



第三题：5.12 证明：任意一棵非空的二叉树的先序序列的最后一个结点一定是叶子结点。

证明：根据先序序列的遍历，二叉树的每一个结点有子节点时，其就不是最后一个节问点，直到某个结点没有子结点，序列终止，此时该结点是叶子结点。

第四题：5.14 设计算法以输出二叉树中先序序列的前 k（k>0）个结点的值。

算法分析：

在先序遍历的基础上，增加判断是否已经输出k个结点。

部分二叉树构造代码略，只附上与此算法有关的代码

#include <iostream>

using namespace *std*;

#include "linkedtree.h"

int main()

{

linkedtree B;

BiNode\* b = new BiNode;

B.createbitree(b);

int i=0,j;

*cin* >> j;

B.outmem(b,i, j);

return 0;

}

void linkedtree::outmem(BiNode\* T,int,int j)

{

static int i = 0;//全局变量用来计数

if (j<=0)

{

*cout* << "请重新输入要输出的结点个数" << *endl*;

}

else if (i<j&&T != *NULL*)

{

//小于要输出的结点个数时，递归调用

*cout* << T->*data*<<" ";

i++;

outmem(T->lChild, i,j);

outmem(T->rChild, i,j);

}

}

第五题：5.16 设计算法以输出每个结点到根结点之间的路径上的所有结点的值。

算法分析：

要查找一个结点，再逆序输出其到根结点之间的结点，因为查找过程中就包含路径的的逆序，所以查找时顺便就将路径存储在数组中，找到结点后逆序输出数组,采用传递实参flag作为标记来判断是否终止递归，结点是否存在。

源代码（部分无关代码没有贴出，只保留函数以及相关部分）

void linkedtree::pathprint1(BiNode\* T, char path[], int pathLength, int \*flag,elementType X)

{

if (T != *NULL*&&\*flag==0)

{

if (T->*data*==X)

//找到要找的结点，输出经过的路径存储的结点，然后将标记改为找到的状态1，终止递归

{

path[pathLength] = T->*data*;

*cout*<<"节点"<<T->*data*<<"到根节点的路径为: ";

for (int i = pathLength; i >= 0; i--)

{

*cout* << path[i];

}

*cout* << *endl*;

\*flag = 1;

}

else

//如果不是要找的值，将路过的结点存起来。递归继续向下寻找

{

path[pathLength++] = T->*data*;

pathprint1(T->lChild ,path, pathLength,flag,X);

pathprint1(T->rChild, path, pathLength,flag,X);

}

}

}

int main()

{

cout<<"通过键盘输入数据"<<endl;

linkedtree A;

BiNode\* a = new BiNode;

A.createbitree(a);

A.firsttravel(a); cout << endl;

int flag = 0;

char path[100] ;

int pathLength=0;

elementType X;

cin >> X;

A.pathprint1(a, path, pathLength, &flag, X);

if (flag == 0)

{

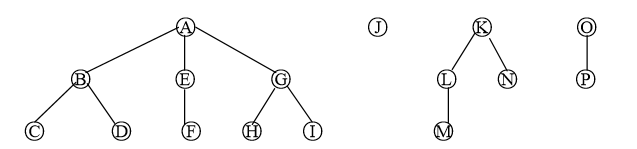
cout << "找不到该结点"<< endl;

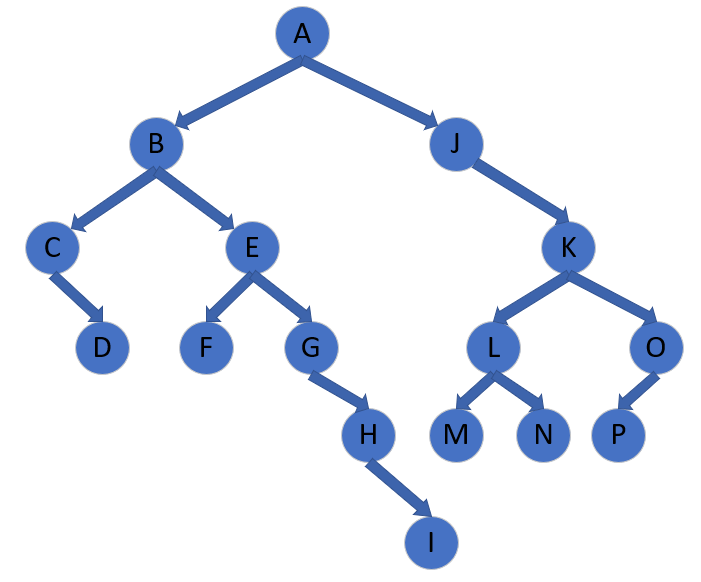
}

return 0;

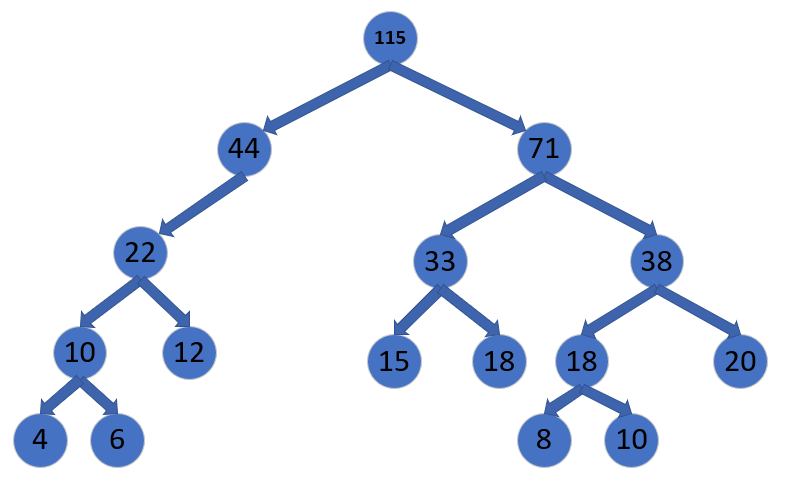
}

第六题：5.23 将下图中的森林转换为对应的二叉树。





第七题：5.33 以数据集合{4,6,8,10,12,15,18,20,22}中的元素为叶子结点的权值构造一棵哈夫曼树，并计算其带权路径长度。



带权路径长度为WPL=4\*(4+6+8+10)+3\*(12+15+18+20)+2\*22=351

第八题：左右子树都存在的二叉树，先序线索化后，存在几个空指针？

**A:0 B:1**

**C:2 D:不确定**

**B**

左右子树均存在，序列最后的叶子结点只有空的左孩子指针指向其前驱，空的右孩子指针没有后继，为空指针，选B。

第九题：左子树不存在的二叉树，先序线索化后存在几个空指针？

**A:0 B:1**

**C:2 D:不确定**

**C**

左子树不存在，该树只有右子树, 根结点为前序序列第一个,没有前驱，根结点的左指针链域就是空的，序列最后的叶子结点，空的右孩子指针没有可指向的后继，因此该结点的右孩子指针链域也是空的,所以选C。

第十题：中序线索化后，存在几个空指针？

**A:0 B:1**

**C:2 D:不确定**

**C**

中序序列第一个结点的左孩子没有可以指向的前驱，为空指针，序列最后的结点右孩子无指向的后继，是空指针,选C。

第十一题：左右子树都存在的二叉树，后序线索化后，存在几个空指针？

**A:0 B:1**

**C:2 D:不确定**

**B**

后序序列中，第一个结点的左孩子的指针无前驱，为空指针，序列最后的根节点的左右孩子均不为空，所以只有一个空指针，选B。

第十二题：右子树不存在的二叉树，后序线索化后，存在几个空指针？

**A:0 B:1**

**C:2 D:不确定**

**C**

右子树不存在，该二叉树只有左支的左子树，后序序列的第一个结点的左子树为空，没有可以指向的前驱，右子树为空有可以指向的后继，序列最后是根结点，根结点的右子树为空，且没有后继，所以为空指针，所以共有两个空指针，选C。