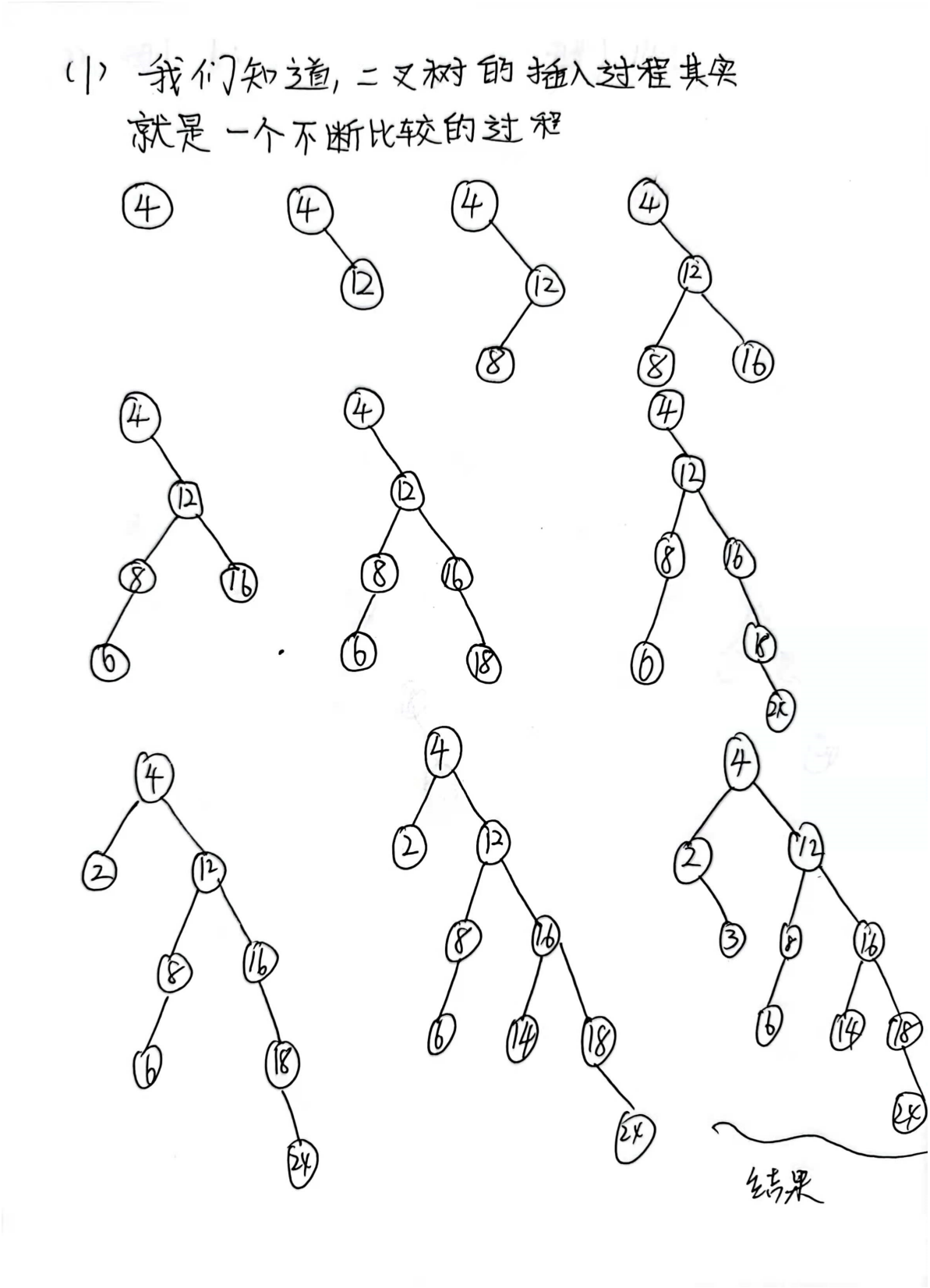
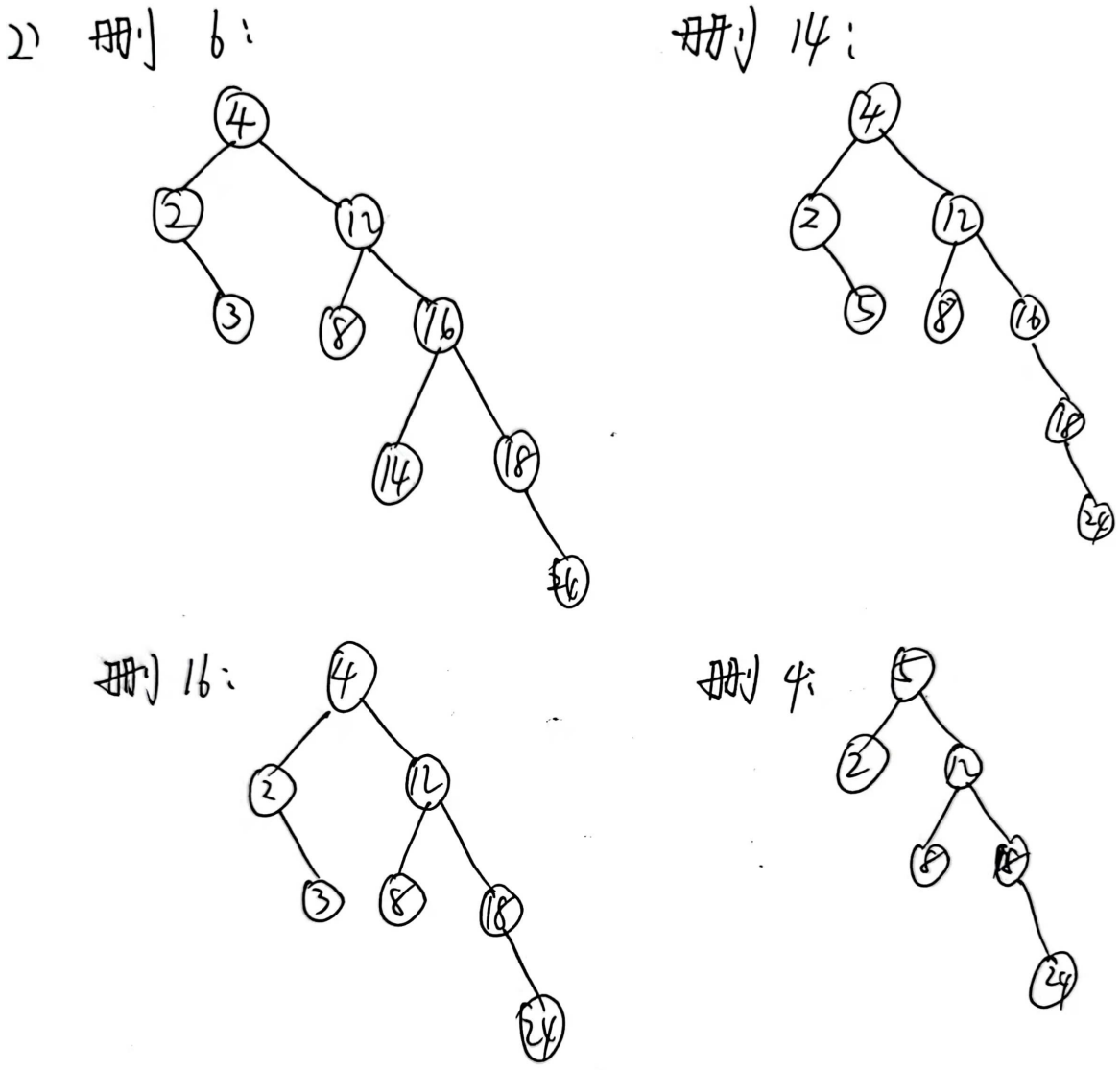
6题

(1)



(2)题



10题

```
template<class K, class E>
void binarySearchTree<K,E>::sort(E*array,int n)
{
    binarySearchTree<K,E>tree;//建一颗空树
    for(int i=0;i<n;i++)
    {
        tree.insert(pair<K,E>(array[i],i));//一个一个插入
    }
    array=inorder_Search(root);//传递指针
}
template<class E>
E* binarySearchTree<K,E>::inorder_Search(binaryTreeNode<E>*ptr)
{
    //本次使用非递归的中序搜寻来实现
    bool *push=new bool [size];
    stack<binaryTreeNode<E>*>s;
    int cnt=0;
    if(root)//根节点非空
    {
        s.push(root);
```

```
        while(!s.empty())
        {
            binaryTreeNode<E>* top=s.top();
            if(!push[top->element])//如果没有被Push进去过
            {
                if(top->leftChild)//存在左孩子
                {
                    s.push(top->leftChild);
                }
                if(top->rightChild)//存在右孩子
                {
                    s.push(top->rightChild);
                }
                push[top->element]=true;//标记为true
            }
            else
            {
                array[cnt++]=top->element;//压入
                s.pop();//弹出，不再访问
            }
        }
    }
    delete []push;
}

```

15题

```
template<class K,class E>
binarySearchTree<K,E>& binarySearchTree<E,K>::deleteMax()
{
    if(!root)
    {
        throw "the tree is empty";//异常处理
    } else
    {
        binaryTreeNode<K>*s=root;//
        binaryTreeNode<K>*fa= nullptr;//是s指针的上一个位置
        while (s)
        {
            fa=s;//保存记录
            s=s->rightChild;
        }
        delete s;//删除
        if(s==root)
        {
            root= nullptr;//如果只有一个元素，那么设置root为空
        }
        fa->rightChild= nullptr;//删除了右结点，所以右孩子应该设置为空
    }
}
```