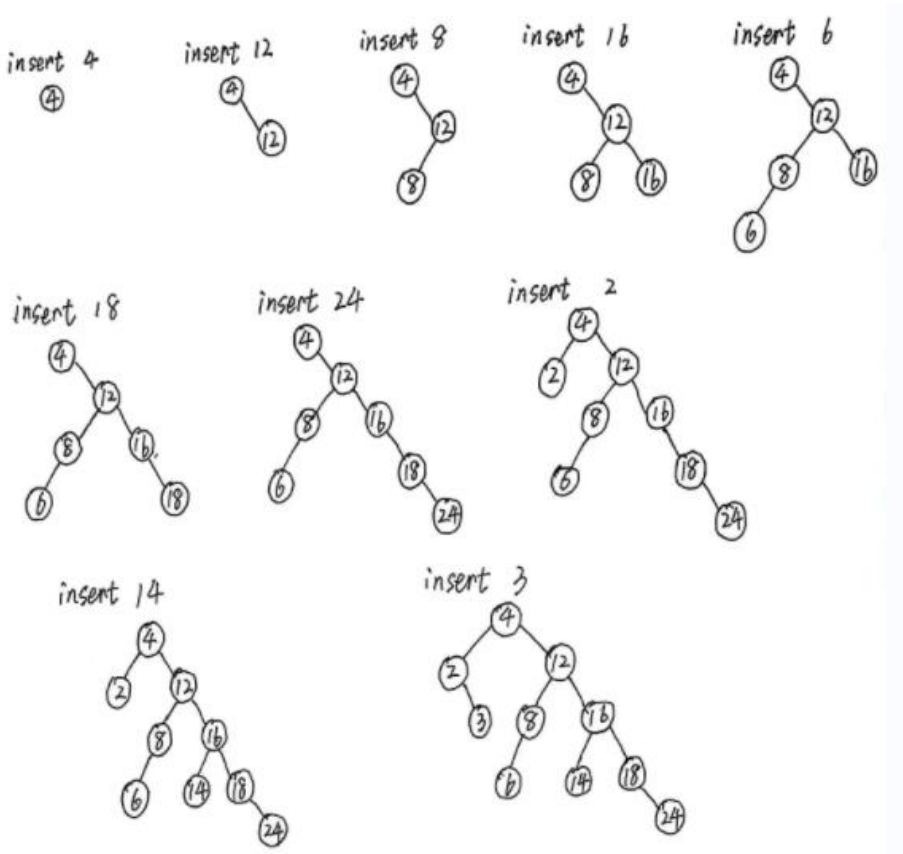
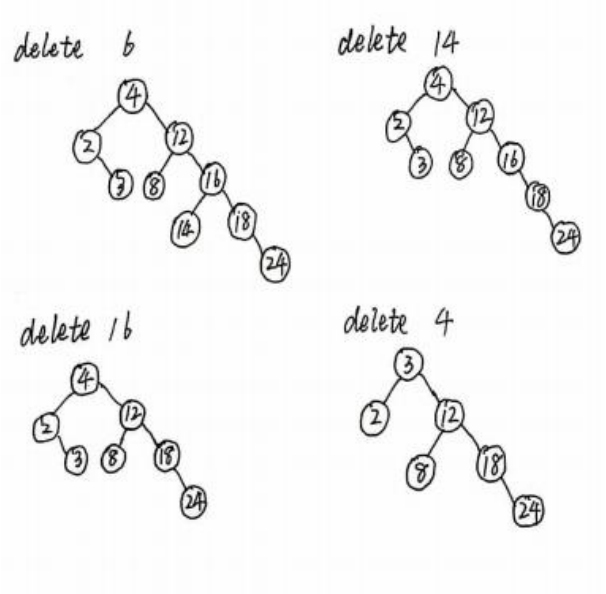
作业 9 参考答案（by 薛正康）

P346-6

（1）



（2）



P346-10

```
//插入函数
template <class T>
void linkedSearchTree<T>::insert(const T& theElement) {
    //p 为当前节点，fp 为 p 节点的父节点
    binaryTreeNode<T> *p = root, *fp = NULL;
    //找到要插入的位置即为 p，theElement 将插入到 fp 的儿子处
    while(p != NULL) {
        fp = p;
        if(theElement < p->element)
            p = p->leftChild;
        else if(theElement > p->element)
            p = p->rightChild;
        else
            return;
    }
    binaryTreeNode<T> *newNode = new binaryTreeNode<T> (theElement);
    //将 theElement 插入，如果树中已有元素则正常插入
    if(root != NULL)
        if(theElement < fp->element)
            fp->leftChild = newNode;
        else fp->rightChild = newNode;
    //否则插入根的位置
    else
        root = newNode;
    //维护树大小
    treeSize++;
}

//执行中序遍历，将结果存入 a 数组中
template <class T>
void linkedSearchTree<T>::do_inOrder(T *a) {
    int cnt = 0;
    inOrder(root, a, cnt);
}

//递归实现中序遍历
template <class T>
void linkedSearchTree<T>::inOrder(binaryTreeNode<T> *t, T *a, int &cnt) {
    if(t != NULL) {
        inOrder(t->leftChild, a, cnt);
        a[cnt++] = t->element;
        inOrder(t->rightChild, a, cnt);
```

```
    }
}

//排序，通过逐个插入并求中序遍历的方式将排序结果存入 a 中
template <class T>
void sort(linkedSearchTree<T> theTree, T *a, int size) {
    for (int i = 0; i < size; i++)
        theTree.insert(a[i]);
    theTree.do_inOrder(a);
    return;
}
```

时间复杂度：

插入排序时间复杂度为 $O(n^2)$；堆排序时间复杂度为 O(nlogn)，其中建堆为 O(n)，每次从堆中提取最小元素为 O(log n)；二叉搜索树最坏情况时间复杂度为 $O(n^2)$，最坏情况在数组有序时出现，整体等效于一个链表，因此可以考虑使用平衡二叉搜索树，时间复杂度将优化至 O(nlogn)。