数据结构与算法分析期末试题 A(2011 级)参考答案

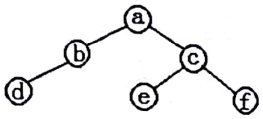
一、单项选择题（每小题 2 分，共 20 分）

1. C)　2. D)　3. B)　4. D)　5. D)

6. B)　7. C)　8. C)　9. D)　10. C)

二、(本题 10 分)

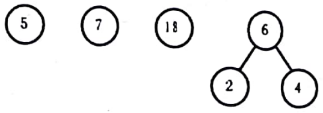
参考答案：



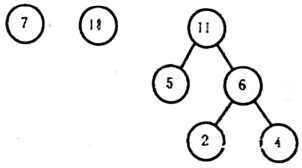
三、(本题 10 分)

参考答案：按照哈夫曼树的算法构造哈夫曼树的过程如下图所示。

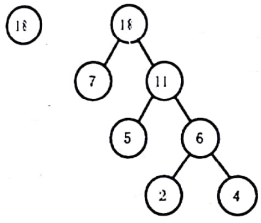
(1)



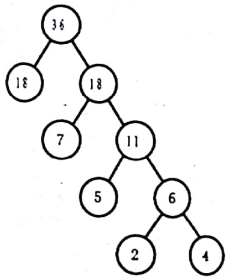
(2)



(3)



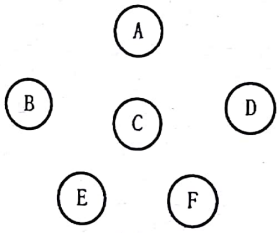
(4)



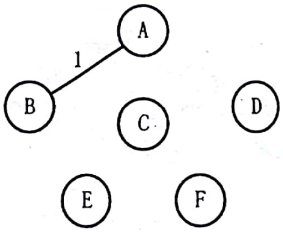
(5)

四、(本题 10 分)

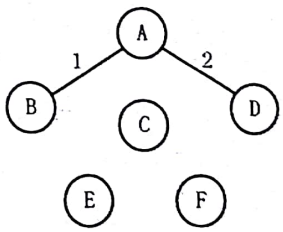
参考答案：Kruskal 的构造顺序如下图所示。



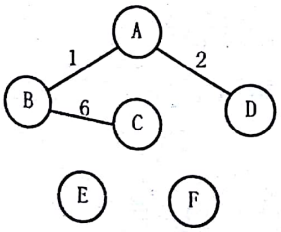
(1)



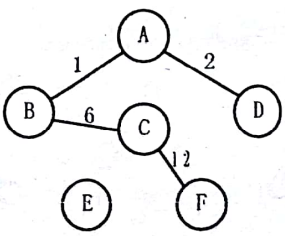
(2)



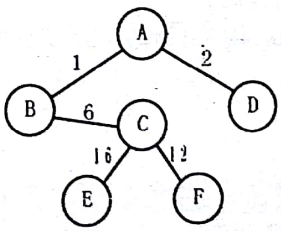
(3)



(4)



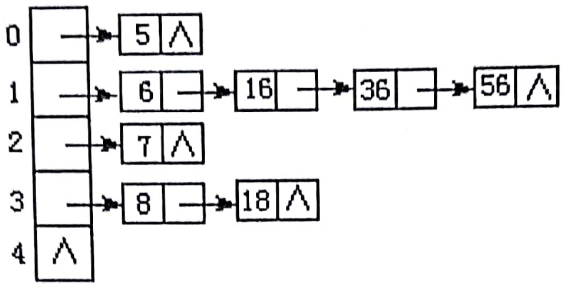
(5)



(6)

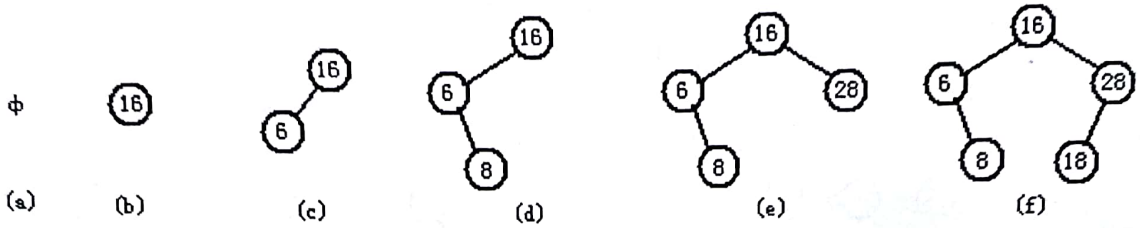
五、(本题 10 分)

参考答案：在用链地址法处理冲突构造哈希表时，查找某关键字的比较次数此关键字在在链表中的序号。哈希表如下图所示：

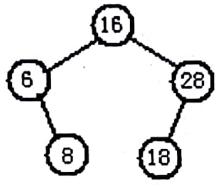


六、（本题 10 分）

参考答案：构造二叉树的过程如下：

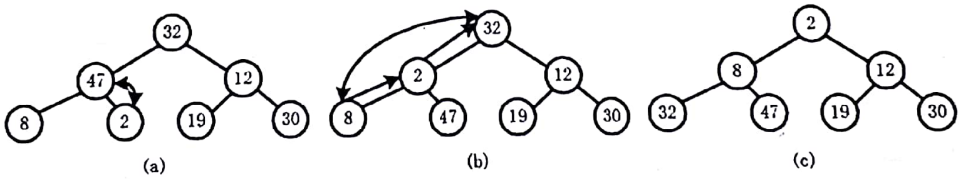


构造的二叉排序树如下图所示：

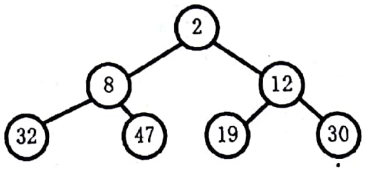


七、（本题 10 分）

参考答案：堆顶元素最小的初始堆构造过程如下图所示。

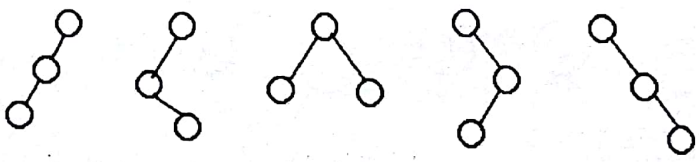


堆顶元素最小的初始堆如下图所示：



八、（本题 10 分）

参考答案：具有 3 个结点的二叉树的不同形态如下图所示。



九、（本题 10 分）

参考答案：将算法实现函数声明为二叉树类的友元函数，可采用层次遍历的方式进行复制，将已复制的结点进入一个队列中即可。

具体算法实现如下：

```
template <class ElemType>
void CopyBitree(BinaryTree<ElemType> *fromBtPtr, BinaryTree<ElemType> *&toBtPtr)
// 操作结果: 复制二叉树 fromBt 到 toBt 的非递归算法
{
    if (toBtPtr != NULL) delete toBtPtr;                    // 释放 toBtPtr
    if (fromBtPtr->Empty())
    {   // 空二叉树
        toBtPtr = NULL;                                     // 空二叉树
    }
    else
    {   // 非空二叉树
        LinkQueue<BinTreeNode<ElemType> *> fromQ, toQ;   // 队列
        BinTreeNode<ElemType> *fromPtr, *toPtr, *fromRoot, *toRoot;
        fromRoot = fromBtPtr->GetRoot();                    // 取出 fromBtPtr 的根
        toRoot = new BinTreeNode<ElemType>(fromRoot->data);     // 复制根结点
        fromQ.InQueue(fromRoot); toQ.InQueue(toRoot);       // 入队
        while (!fromQ.Empty())
        {   // fromQ 非空
            fromQ.OutQueue(fromPtr);                        // 出队
            toQ.OutQueue(toPtr);                            // 出队
            if (fromPtr->leftChild != NULL)
            {   // 左子树非空
                toPtr->leftChild = new BinTreeNode<ElemType>(fromPtr->leftChild->data);
                    // 复制 fromPtr 左孩子
                fromQ.InQueue(fromPtr->leftChild); toQ.InQueue(toPtr->leftChild);   // 入队
            }
            if (fromPtr->rightChild != NULL)
            {   // 右子树非空
                toPtr->rightChild = new BinTreeNode<ElemType>(fromPtr->rightChild->data);
                    // 复制 fromPtr 左孩子
                fromQ.InQueue(fromPtr->rightChild); toQ.InQueue(toPtr->rightChild); // 入队
            }
        }
        toBtPtr = new BinaryTree<ElemType>(toRoot);         // 生成 toBtPtr
    }
}
```