19

武汉大学

2007 年攻读硕士学位研究生入学考试试题

科目名称：数据结构与算法基础　　科目代码：820

注意：　所有的答题内容必须答在答题纸上，凡答在试题或草稿纸上的一律无效

一 填空(每空 3 分，共计 15 分)

1、在一个长度为 n 的顺序存储线性表中，向第 i 个元素(1≤i≤n)之前插入一个新元素时，需要从后向前依次后移_____个元素。

2、由权值分别为 3, 8, 6, 2, 5 的叶子结点生成一棵哈夫曼树，它的带权路径长度为_____。

3、按先根次序法遍历的树林正好等同于按_____遍历相应的二叉树。

4、对于一个具有 n 个顶点和 e 条边的连通图，其生成树中的顶点数和边数分别为_____和_____。

二 问答题（共计 5+5+20+19+15＝64 分）

1、
```
void AE(Stack& S)
{
    int i, x, a[4]={5,8,12,15};
    InitStack(S);
    Push(S,30);
    Push(S,40);
    Push(S,50);
    x=Pop(S)+2*Pop(S);
    Push(S,x);
    for(i=0;i<4;i++) Push(S,a[i]);
    while(!StackEmpty(S))
        printf("%5d", Pop(S));
}
```
写出该算法被调用后得到的输出结果。(5 分)

2、说明下面算法中递归过程所实现的功能（5 分）


```
int unknown( BinTreeNode  * t)
{     //指针 t 是二叉树的根指针。
    if ( t == NULL)    return(-1);
    else if ( unknown(t->leftChild) >= unknown(t->rightChild) )
        return (1+unknown(t->leftChild));
    else
        return (1+unkuown(t->rightChild));
}
```
递归过程的功能：_____。

3、已知一个图的顶点集 V 和边集 G 分别为：

V={0,1,2,3,4,5,6,7};

E={(0,1)8,(0,2)5,(0,3)2,(1,5)6,(2,3)25,(2,4)13,(3,5)9,(3,6)10,(4,6)4,(5,7)20};

①画出上述表示所对应的无向图；

②画出该无向图所对应的邻接表存储结构示意图；

③针对②的存储结构，写出它的深度优先遍历序列和广度优先遍历序列。

④采用普里姆算法构造最小生成树，试写出构造过程中依次得到的各条边；

⑤采用克鲁斯卡尔算法构造最小生成树，试写出构造过程中依次得到的各条边。（20 分）

4、已知待排序文件各记录的排序码顺序如下：T={12，2，16，30，8，28，4，10，20，6，18}，请列出用下列算法对该序列做升序排序过程中，第一趟的排序结果。

①希尔排序（第一趟排序的增量为 5 ）(6 分)

②快速排序（选第一个记录为枢轴（分隔））(6 分)

③链式基数排序（基数为 1 0 ）(7 分)

5、设有一组待散列存储的关键字序列为{32，75，29，63，48，94，25，46，18，70}，其散列地址空间为 HT[13]（即散列地址空间为 0~12），若采用除留余数法构造哈希函数并采用开放地址的线性探测再散列法解决冲突，试在散列地址空间中对元素序列构造哈希表，并画出最后得到的哈希表，求出平均查找长度。（15 分）

三 算法设计题（共计 20+15+15＝50 分）

1、已知一棵二叉树的先序遍历和中序遍历序列分别在于两个一维数组中，试编写算法建立该二叉树（要求采用链接存储结构进行存储），并给出对该二叉树进行后序遍历的非递归算法。（20 分）

2、编写一算法，求出一棵二叉树中所有结点数和叶子结点数，假定分别用变参 C1 和 C2 分别统计所有结点数和叶子结点数，初值均为 0。

void Count(BTreeNode *BT, int &C1, int &C2) （15 分）

3、编写向类型为 List 的线性表 L 中第 i 个元素位置插入一个元素的算法，假定不需要对 i 的值进行有效性检查，同时不需要检查存储空间是否用完。

void Insert(List& L, int i, ElemType x ）（15 分）

四 程序填充题（每空 3 分，共计 21 分）

1、在串的堆分配存储结构上实现串的联接算法。

```
status  Concat ( HString S1, HString S2, HString &T)
{   //用 T 返回由 S1 和 S2 联接而成的新串
    if ( T.ch )  free( T.ch );          //释放旧空间
    if (!(T.ch =_____________________))
      exit(overflow);
    for ( i=S1.length-1; i>=0; --i)
      T.ch[i] = S1.ch[i];
    _____________________;
    for ( i=T.length-1; i>=S1.length; --i)
      _____________________;
    return ok;
}
```

2、在二叉排序树中删除指定值域的结点


```
void delete( BTree **BT, int x)
{
   BTree *p=*BT, *q, *r, *t;
   q = NULL;      //p 指向待比较的结点,q 指向 p 的前驱结点

   while (p!=NULL && p->data!=x)
   {
        if ( x < p->data )
           {q = p; p = p->left;}
        else
           {q = p; p = p->right;}
   }
   if ( p == NULL )
        printf("未发现数据域为%d 的结点\n",x);
   else if ( p->left == NULL )          //被删结点无左子树
   {
        if (q == NULL)
           t = p->right;
        else if (q->left == p)
           q->left =p->right;
        else
           _____________________;
   }
   else          //被删结点有左子树
   {             //查找被删结点的左子树之最右结点
        _____________________;
        while (_____________________)
           r=r->right;
        r->right = p->right; /*被删结点的右子树作为 r 的右子树*/
        if (q == NULL)
           t = p->left;
        else if (q->left == p)
           _____________________;
        Else
           q->right = p->left;
   }
}
```