杭州电子科技大学

2013 年攻读硕士学位研究生入学考试

《数据结构》试题

（试题共 六 大题， 6 页，150 分）

姓名_____报考专业_____准考证号_____

【所有答案必须写在答题纸上，做在试卷或草稿纸上无效！】

一、是非题（每小题 2 分，共 10 分）

1. 对于插入、删除操作，单链表和顺序表的时间复杂度都可计为 O（n）。
2. 队列是一种操作受限的线性表，所有对数据元素的操作仅限一端进行。
3. 非线性结构的遍历过程是对结构中的每一数据元素访问且仅访问一次，与结构中数据元素之间的关系无关。
4. 对于求最小代价生成树的方法，Kruskal 方法优于 Prim 方法。
5. 哈希表的查找效率与查找表的长度无关。

二、选择题（每空 2 分，共 28 分）

1. 线性表在_____的情况下适于使用链表结构实现。
   a：表中含有大量结点　　b：需经常修改表中结点值
   c：需经常对表进行删除、插入　　d：表中数据元素依关键字有序

2. 如下关于串的陈述中，错误的是_____。
   a：串是数据对象为字符集的线性表　　b：串的长度为字符的个数
   c：串中若干个连续字符构成的子序列称为子串　　d：串中的数据元素是字母

3. 设有二维数组 A[6][5]，每一数组元素所用存储空间为 4 个字节，存储器按字节编址。已知 A 在存储器中的起始地址为 100，则按行存储时，元素 A[2][3]的第一个字节的地址是_____；按列存储时，元素 A[2][3]的第一个字节的地址是_____。
   a：128　　b：152　　c：180　　d：200

4. 对广义表 A=(((a, b), (c)), (d))
   执行操作 gettail(gethead (gettail(A)))的结果是：_____。
   执行操作 gethead(gettail(gethead(A)))的结果是：_____。
   a：( )　　b：(a)　　c：(b)　　d：(c)　　e：(d)

5. 压栈次序为 1、2、3、4 则不可能得到的输出序列是_____。
   a: 1432　　b: 2413　　c: 3421　　d: 4321

6. 设任意无向图 G = (V, E)和 G' = (V', E')，若 G' 是 G 的连通分量，则下列说法中不正确的是_____。
   a: G' 是 G 的子图　　b: G' 是 G 的连通子图
   c: G' 是 G 的极大连通子图　　d: G' 是 G 的极大连通子图且 V' 等于 V

7. 下列查找方法中_____针对关键字有序的顺序表查找效率较低。
   a: 顺序查找　　b: 折半查找　　c: 分块查找　　d: 斐波那契查找

8. 下列排序方法中，_____是稳定的；_____ 具有最好的平均性能；_____所需的辅存空间最大。面对不同的初始数据，_____的时间复杂度恒为 O (n log n)；而_____的时间复杂度恒为 O (n^2)；
   a: 快速排序　　b: 简单选择排序　　c: 希尔排序　　d:归并排序

三、填空题（每空 2 分，共 22 分）

1. 以物理位置来表示数据元素之间的逻辑关系的存储结构被称为_____；
   通过指针来保持数据元素之间的逻辑关系的存储结构被称为_____；

2. 对完全二叉树中的结点从 1 开始按层进行连续编号。设编号为 i 的结点的父结点存在，则编号为_____ 的结点为其父结点；设编号为 i 的左孩子结点存在，则编号为 _____ 的结点为其左孩子结点；设编号为 i 的右孩子结点存在，则编号为 _____ 的结点为其右孩子结点。

3. 已知某二叉树的先序遍历次序为：A, B, C, D, E, F, G, H。中序遍历次序为：B, D, C, F, E, A, H, G。则其后序遍历次序为_____；层次遍历次序为_____。

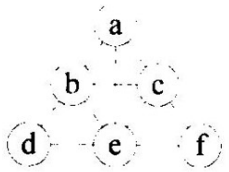
4. n 个顶点的强连通图至少有_____条弧，至多有_____条弧。

5. 一棵 m 阶的 B 树，第一层至少有一个结点；第二层至少有2个结点，除根之外的所有非终端结点至少有_____棵子树，树中所有非终端结点至多有_____棵子树。

四、图示结构题（每小题 8 分，共 40 分）

1. 已知某森林的先序遍历次序为：A, D, E, F, G, H, B, I, C, J, K, L, M, N。
   中序遍历次序为：D, F, G, H, E, A, I, B, J, L, M, N, K, C。
   （1）　画出该森林。
   （2）　画出该森林用孩子兄弟法表示的存储结构。

2. 已知某无向图如右图所示：



（1） 画出该图数组表示法（邻接矩阵）存储结构。
（2） 画出该图的邻接表存储结构。
（3） 根据你所绘制的邻接表给出 DFS 及 BFS 次序，并画出深度优先生成树和广深度优先生成树。

3. 依序将关键字 65, 50, 30, 20, 10, 45, 60, 55, 25, 70 插入到一棵二叉排序树中（初始状态为空）。
（1） 请画出该二叉排序树。
（2） 若之后删除关键字 65，画出删除后的二叉排序树。
（3） 以同样的关键字插入次序建立平衡二叉排序树，请画出该平衡二叉树。

4. 设哈希函数为 H(key)=key mod 13，哈希表长为 15，用开放定址法处理冲突，增量序列使用二次探测再散列。若依次在哈希表中插入 11 个元素：
34, 12, 67, 43, 98, 23, 51, 86, 05, 37, 22。
（1） 画出它们在表中的分布情形。
（2） 求其等概率情况下平均成功的查找长度。

5. 假设用于通讯的电文仅由 8 个字符 A, B, C, D, E, F, G, H 组成，字符在电文中出现的频率分别为 6, 25, 3, 17, 8, 15, 10, 16
（1） 画出你所建的哈夫曼树，
（2） 给出每一字符的哈夫曼编码。

五、阅读以下函数，指出算法的功能（每小题 6 分，共 30 分）

1.
```
bool A1(LinkList L, int i, ElemType e)
{ // L 为带头结点的单链表
    int j=0;
    LinkList  p = L, s;
    while(p && j < i-1) {
        p=p->next;
        j++;
    }
    if(!p||j>i-1)      return ERROR;
    s=(LinkList)malloc(sizeof(LNode));
    s->data=e;
    s->next=p->next;
    p->next=s;
    return OK;
}
```

2.
```
void  A2(LinkQueue &Q)
{ // Q 为带头结点的链队列、Q.front、 Q.rear 分别为队头和队尾指针
    Qnode *p,*q;
    Q.rear = Q.front;
    P = Q.front->next;
    Q.front->next = NULL;
    while(p){
        q=p;
        p=p->next;
        free(q);
    }
}
```

3.
```
int A3(BiTree T)
{ // T 为二叉链表表示的二叉树
    int i,j;
    if(!T)   return 0;
    if(T->lchild)
        i=A3(T->lchild);
    else
        i=0;
    if(T->rchild)
        j=A3(T->rchild);
    else
        j=0;
    return i>j?i+1:j+1;
}
```

4.
```
int A4(SSTable ST,KeyType key)
{ // ST 为顺序表，表中数据元素依关键字有序
    int low,high,mid;
    low=0;
    high = ST.length-1;
    while(low <= high){
        mid=(low+high)/2;
        if (key == ST.elem[mid].key)
            return  mid;
        else if(key < ST.elem[mid].key)
            high=mid-1;
```

```
            else
                low=mid+1;
        }
        return -1;
    }
```

5.
```
int A5(SqList &L, int low, int high)
{ //L 为顺序表。其中：下标为 0 的单元未存数据。
    KeyType  pivotkey;
    L.elem[0] = L. elem [low];
    Pivotkey = L.elem [low].key;
    while(low< high){
        while(low<high && L.elem [high].key >= Pivotkey)
            --high;
        L.elem [low]=L.elem [high];
        while(low < high && L.elem [low].key <= Pivotkey)
            ++low;
        L.elem [high]=L elem [low];
    }
    L.elem[low]=L.elem [0];
    return low;
 }
```

六、算法设计题（每小题 10 分，共 20 分）

1. 已知集合A和集合B存在，且分别用带头结点的单链表L1和L2表示。
编写算法：Diff(LinkList &L1, LinkList L2)
实现集合运算：A = A - B。
设单链表结点结构为：

```
typedef  struct  LNode {
    ElemType    data;
    struct  LNode  next;
}LNode, *LinkList;
```

2. 设树的存储结构为孩子链表表示法。其中：

孩子结点结构为：

```
typedef struct CTNode {
        int          child;
        struct CTNode *next;
} *ChildPtr;
```

双亲结点结构为：

```
typedef struct {
        ElemType  data;
        ChildPtr  firstchild;  // 孩子链的头指针
} CTBox;
```

树结构为：

```
typedef struct {
       CTBox  nodes[MAX_TREE_SIZE];
       int    n, r;    // 结点数和根结点的位置
}CTree;
```

试编写一递归的先根遍历树的算法。

```
POTT( CTree  &T, int v, void( *visit)(ElemType  e))
//已知树T非空，v的初值为 T.r（根结点的位置），visit为访问函数。
```