**数据结构试卷（七）**

**一、选择题(30分)**

1．设某无向图有n个顶点，则该无向图的邻接表中有（ ）个表头结点。

(A) 2n (B) n (C) n/2 (D) n(n-1)

2．设无向图G中有n个顶点，则该无向图的最小生成树上有（ ）条边。

(A) n (B) n-1 (C) 2n (D) 2n-1

3．设一组初始记录关键字序列为(60，80，55，40，42，85)，则以第一个关键字45为基准而得到的一趟快速排序结果是（ ）。

(A) 40，42，60，55，80，85 (B) 42，45，55，60，85，80

(C) 42，40，55，60，80，85 (D) 42，40，60，85，55，80

4．（ ）二叉排序树可以得到一个从小到大的有序序列。

(A) 先序遍历 (B) 中序遍历 (C) 后序遍历 (D) 层次遍历

5．设按照从上到下、从左到右的顺序从1开始对完全二叉树进行顺序编号，则编号为i结点的左孩子结点的编号为（ ）。

(A) 2i+1 (B) 2i (C) i/2 (D) 2i-1

6．程序段s=i=0；do {i=i+1； s=s+i；}while(i<=n)；的时间复杂度为（ ）。

(A) O(n) (B) O(nlog2n) (C) O(n2) (D) O(n3/2)

7．设带有头结点的单向循环链表的头指针变量为head，则其判空条件是（ ）。

(A) head==0 (B) head->next==0

(C) head->next==head (D) head!=0

8．设某棵二叉树的高度为10，则该二叉树上叶子结点最多有（ ）。

(A) 20 (B) 256 (C) 512 (D) 1024

9．设一组初始记录关键字序列为(13，18，24，35，47，50，62，83，90，115，134),则利用二分法查找关键字90需要比较的关键字个数为（ ）。

(A) 1 (B) 2 (C) 3 (D) 4

10.设指针变量top指向当前链式栈的栈顶，则删除栈顶元素的操作序列为（ ）。

(A) top=top+1; (B) top=top-1;

(C) top->next=top; (D) top=top->next;

**二、判断题(20分)**

1．不论是入队列操作还是入栈操作，在顺序存储结构上都需要考虑“溢出”情况。（ ）

2．当向二叉排序树中插入一个结点，则该结点一定成为叶子结点。（ ）

3．设某堆中有n个结点，则在该堆中插入一个新结点的时间复杂度为O(log2n)。（ ）

4．完全二叉树中的叶子结点只可能在最后两层中出现。（ ）

5．哈夫曼树中没有度数为1的结点。（ ）

6．对连通图进行深度优先遍历可以访问到该图中的所有顶点。（ ）

7．先序遍历一棵二叉排序树得到的结点序列不一定是有序的序列。（ ）

8．由树转化成二叉树，该二叉树的右子树不一定为空。（ ）

9．线性表中的所有元素都有一个前驱元素和后继元素。（ ）

10.带权无向图的最小生成树是唯一的。（ ）

**三、填空题(30分)**

1. 1.        设指针变量p指向双向链表中的结点A，指针变量s指向被插入的结点X，则在结点A的后面插入结点X的操作序列为\_\_\_\_\_\_\_\_\_=p；s->right=p->right；\_\_\_\_\_\_\_\_\_\_=s； p->right->left=s；（设结点中的两个指针域分别为left和right）。
2. 2.        设完全有向图中有n个顶点，则该完全有向图中共有\_\_\_\_\_\_\_\_条有向条；设完全无向图中有n个顶点，则该完全无向图中共有\_\_\_\_\_\_\_\_条无向边。
3. 3.        设关键字序列为(Kl，K2，…，Kn)，则用筛选法建初始堆必须从第\_\_\_\_\_\_个元素开始进行筛选。
4. 4.        解决散列表冲突的两种方法是\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_和\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。
5. 5.        设一棵三叉树中有50个度数为0的结点，21个度数为2的结点，则该二叉树中度数为3的结点数有\_\_\_\_\_\_个。
6. 6.        高度为h的完全二叉树中最少有\_\_\_\_\_\_\_\_个结点，最多有\_\_\_\_\_\_\_\_个结点。
7. 7.        设有一组初始关键字序列为(24，35，12，27，18，26)，则第3趟直接插入排序结束后的结果的是\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。
8. 8.        设有一组初始关键字序列为(24，35，12，27，18，26)，则第3趟简单选择排序结束后的结果的是\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。
9. 9.        设一棵二叉树的前序序列为ABC，则有\_\_\_\_\_\_\_\_\_\_\_\_\_\_种不同的二叉树可以得到这种序列。
10. 10.     下面程序段的功能是实现一趟快速排序，请在下划线处填上正确的语句。

struct record {int key;datatype others;};

void quickpass(struct record r[], int s, int t, int &i)

{

int j=t; struct record x=r[s]; i=s;

while(i<j)

{

while (i<j && r[j].key>x.key) j=j-1; if (i<j) {r[i]=r[j];i=i+1;}

while (\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_) i=i+1; if (i<j) {r[j]=r[i];j=j-1;}

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;

}

**四、算法设计题(20分)**

1. 1.         设计在链式结构上实现简单选择排序算法。
2. 2.         设计在顺序存储结构上实现求子串算法。
3. 3.         设计求结点在二叉排序树中层次的算法。

**数据结构试卷（七）**

**一、选择题**

1．B 2．B 3．C 4．B 5．B

6．A 7．C 8．C 9．B 10．D

**二、判断题**

1．对 2．对 3．对 4．对 5．对

6．对 7．对 8．错 9．错 10．错

**三、填空题**

1. 1.         s->left=p，p->right
2. 2.         n(n-1)，n(n-1)/2
3. 3.         n/2
4. 4.         开放定址法，链地址法
5. 5.         14
6. 6.         2h-1，2h-1
7. 7.         (12，24，35，27，18，26)
8. 8.         (12，18，24，27，35，26)
9. 9.         5
10. 10.     i<j && r[i].key<x.key，r[i]=x

**四、算法设计题**

1. 1.         设计在链式结构上实现简单选择排序算法。

void simpleselectsorlklist(lklist \*&head)

{

lklist \*p,\*q,\*s; int min,t;

if(head==0 ||head->next==0) return;

for(q=head; q!=0;q=q->next)

{

min=q->data; s=q;

for(p=q->next; p!=0;p=p->next) if(min>p->data){min=p->data; s=p;}

if(s!=q){t=s->data; s->data=q->data; q->data=t;}

}

}

1. 2.         设计在顺序存储结构上实现求子串算法。

void substring(char s[ ], long start, long count, char t[ ])

{

long i,j,length=strlen(s);

if (start<1 || start>length) printf("The copy position is wrong");

else if (start+count-1>length) printf("Too characters to be copied");

else { for(i=start-1,j=0; i<start+count-1;i++,j++) t[j]=s[i]; t[j]= '\0';}

}

1. 3.         设计求结点在二叉排序树中层次的算法。

int lev=0;

typedef struct node{int key; struct node \*lchild,\*rchild;}bitree;

void level(bitree \*bt,int x)

{

if (bt!=0)

{lev++; if (bt->key==x) return; else if (bt->key>x) level(bt->lchild,x); else level(bt->rchild,x);}

}