**数据结构试卷（十）**

**一、选择题(24分)**

1．下列程序段的时间复杂度为（ ）。

i=0，s=0； while (s<n) {s=s+i；i++；}

(A) O(n1/2) (B) O(n1/3) (C) O(n) (D) O(n2)

2．设某链表中最常用的操作是在链表的尾部插入或删除元素，则选用下列（ ）存储方式最节省运算时间。

(A) 单向链表 (B) 单向循环链表

(C) 双向链表 (D) 双向循环链表

3．设指针q指向单链表中结点A，指针p指向单链表中结点A的后继结点B，指针s指向被插入的结点X，则在结点A和结点B插入结点X的操作序列为（ ）。

(A) s->next=p->next；p->next=-s； (B) q->next=s； s->next=p；

(C) p->next=s->next；s->next=p； (D) p->next=s；s->next=q；

4．设输入序列为1、2、3、4、5、6，则通过栈的作用后可以得到的输出序列为（ ）。

(A) 5，3，4，6，1，2 (B) 3，2，5，6，4，1

(C) 3，1，2，5，4，6 (D) 1，5，4，6，2，3

5．设有一个10阶的下三角矩阵A（包括对角线），按照从上到下、从左到右的顺序存储到连续的55个存储单元中，每个数组元素占1个字节的存储空间，则A[5][4]地址与A[0][0]的地址之差为（ ）。

(A) 10 (B) 19 (C) 28 (D) 55

6．设一棵m叉树中有N1个度数为1的结点，N2个度数为2的结点，……，Nm个度数为m的结点，则该树中共有（ ）个叶子结点。

(A)  (B)  (C)  (D) 

7. 二叉排序树中左子树上所有结点的值均（ ）根结点的值。

(A) < (B) > (C) = (D) !=

8. 设一组权值集合W=(15，3，14，2，6，9，16，17)，要求根据这些权值集合构造一棵哈夫曼树，则这棵哈夫曼树的带权路径长度为（ ）。

(A) 129 (B) 219 (C) 189 (D) 229

9. 设有n个关键字具有相同的Hash函数值，则用线性探测法把这n个关键字映射到HASH表中需要做（ ）次线性探测。

(A) n2 (B) n(n+1) (C) n(n+1)/2 (D) n(n-1)/2

10.设某棵二叉树中只有度数为0和度数为2的结点且度数为0的结点数为n，则这棵二叉中共有（ ）个结点。

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

11.设一组初始记录关键字的长度为8，则最多经过（ ）趟插入排序可以得到有序序列。

(A) 6 (B) 7 (C) 8 (D) 9

12.设一组初始记录关键字序列为(Q，H，C，Y，P，A，M，S，R，D，F，X)，则按字母升序的第一趟冒泡排序结束后的结果是（ ）。

(A) F，H，C，D，P，A，M，Q，R，S，Y，X

(B) P，A，C，S，Q，D，F，X，R，H，M，Y

(C) A，D，C，R，F，Q，M，S，Y，P，H，X

(D) H，C，Q，P，A，M，S，R，D，F，X，Y

**二、填空题(48分，其中最后两小题各6分)**

1. 设需要对5个不同的记录关键字进行排序，则至少需要比较\_\_\_\_\_\_\_\_\_\_\_\_\_次，至多需要比较\_\_\_\_\_\_\_\_\_\_\_\_\_次。
2. 快速排序算法的平均时间复杂度为\_\_\_\_\_\_\_\_\_\_\_\_，直接插入排序算法的平均时间复杂度为\_\_\_\_\_\_\_\_\_\_\_。
3. 设二叉排序树的高度为h，则在该树中查找关键字key最多需要比较\_\_\_\_\_\_\_\_\_次。
4. 设在长度为20的有序表中进行二分查找，则比较一次查找成功的结点数有\_\_\_\_\_\_\_\_\_个，比较两次查找成功有结点数有\_\_\_\_\_\_\_\_\_个。
5. 设一棵m叉树脂的结点数为n，用多重链表表示其存储结构，则该树中有\_\_\_\_\_\_\_\_\_个空指针域。
6. 设指针变量p指向单链表中结点A，则删除结点A的语句序列为：

q=p->next；p->data=q->data；p->next=\_\_\_\_\_\_\_\_\_\_\_；feee(q)；

1. 数据结构从逻辑上划分为三种基本类型：\_\_\_\_\_\_\_\_\_\_\_、\_\_\_\_\_\_\_\_\_\_和\_\_\_\_\_\_\_\_\_\_\_。
2. 设无向图G中有n个顶点e条边，则用邻接矩阵作为图的存储结构进行深度优先或广度优先遍历时的时间复杂度为\_\_\_\_\_\_\_\_\_；用邻接表作为图的存储结构进行深度优先或广度优先遍历的时间复杂度为\_\_\_\_\_\_\_\_\_。
3. 设散列表的长度为8，散列函数H(k)=k % 7，用线性探测法解决冲突，则根据一组初始关键字序列(8，15，16，22，30，32)构造出的散列表的平均查找长度是\_\_\_\_\_\_\_\_。
4. 设一组初始关键字序列为(38，65，97，76，13，27，10)，则第3趟冒泡排序结束后的结果为\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。
5. 设一组初始关键字序列为(38，65，97，76，13，27，10)，则第3趟简单选择排序后的结果为\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。
6. 设有向图G中的有向边的集合E={<1，2>，<2，3>，<1，4>，<4，5>，<5，3>，<4，6>，<6，5>}，则该图的一个拓扑序列为\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。
7. 下面程序段的功能是建立二叉树的算法，请在下划线处填上正确的内容。

typedef struct node{int data;struct node \*lchild;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;}bitree;

void createbitree(bitree \*&bt)

{

scanf(“%c”,&ch);

if(ch=='#') \_\_\_\_\_\_\_\_\_\_\_;else

{ bt=(bitree\*)malloc(sizeof(bitree)); bt->data=ch; \_\_\_\_\_\_\_\_;createbitree(bt->rchild);}

}

1. 下面程序段的功能是利用从尾部插入的方法建立单链表的算法，请在下划线处填上正确的内容。

typedef struct node {int data; struct node \*next;} lklist;

void lklistcreate(\_\_\_\_\_\_\_\_\_\_\_\_\_ \*&head )

{

for (i=1;i<=n;i++)

{

p=(lklist \*)malloc(sizeof(lklist));scanf(“%d”,&(p->data));p->next=0;

if(i==1)head=q=p;else {q->next=p;\_\_\_\_\_\_\_\_\_\_\_\_;}

}

}

**三、算法设计题(22分)**

* 1. 设计在链式存储结构上合并排序的算法。
  2. 设计在二叉排序树上查找结点X的算法。
  3. 设关键字序列(k1，k2，…，kn-1)是堆，设计算法将关键字序列(k1，k2，…，kn-1，x)调整为堆。

**数据结构试卷（十）参考答案**

**一、选择题**

1．A 2．D 3．B 4．B 5．B 6．D

7．A 8．D 9．D 10．C 11．B 12．D

**二、填空题**

1. 4，10
2. O(nlog2n)，O(n2)
3. n
4. 1，2
5. n(m-1)+1
6. q->next
7. 线性结构，树型结构，图型结构
8. O(n2)， O(n+e)
9. 8/3
10. (38，13，27，10，65，76，97)
11. (10，13，27，76，65，97，38)
12. 124653
13. struct node \*rchild，bt=0，createbitree(bt->lchild)
14. lklist，q=p

**三、算法设计题**

1. 设计在链式存储结构上合并排序的算法。

void mergelklist(lklist \*ha,lklist \*hb,lklist \*&hc)

{

lklist \*s=hc=0;

while(ha!=0 && hb!=0)

if(ha->data<hb->data){if(s==0) hc=s=ha; else {s->next=ha; s=ha;};ha=ha->next;}

else {if(s==0) hc=s=hb; else {s->next=hb; s=hb;};hb=hb->next;}

if(ha==0) s->next=hb; else s->next=ha;

}

1. 设计在二叉排序树上查找结点X的算法。

bitree \*bstsearch1(bitree \*t, int key)

{

bitree \*p=t;

while(p!=0) if (p->key==key) return(p);else if (p->key>key)p=p->lchild; else p=p->rchild;

return(0);

}

1. 设关键字序列(k1，k2，…，kn-1)是堆，设计算法将关键字序列(k1，k2，…，kn-1，x)调整为堆。

void adjustheap(int r[ ],int n)

{

int j=n,i=j/2,temp=r[j-1];

while (i>=1) if (temp>=r[i-1])break; else{r[j-1]=r[i-1]; j=i; i=i/2;}

r[j-1]=temp;

}