**《数据结构》期中试卷 （2009级） 2010-2011学年第一学期**

**姓名： 学号： 成绩：**

1. **选择题：（每小题2分，共20分）**

**1.** 有六个元素6，5，4，3，2，1 的顺序进栈，下列哪一个不是合法的出栈序列？（ ）

A. 5 4 3 6 1 2 B. 4 5 3 1 2 6 C. 3 4 6 5 2 1 D. 2 3 4 1 5 6

**2.** 在一个有125个元素的顺序表中插入一个新元素并保持原来顺序不变，平均要移动（ ）个元素。

A．8 B. 62.5 C. 62 D. 7

**3.** 已知广义表A＝((a,b,c),(d,e,f),(h,(i,j)),g)，从A表中取出原子项e的运算是：( )

A.head(tail(A))　　　B.head(tail(tail(A)))

C.head(head(tail(tail(A))))　　D.head(tail(head(tail(A))))

**4.** 循环队列存储在数组A[0..m]中，设front和rear分别为队列的头指针和尾指针，则入队时的操作为（ ）。

A. front=( front +1) mod (m+1) B. rear=(rear+1) mod (m+1)

C. front=( front +1) mod m D. rear=(rear+1) mod m

5. 在双向循环链表中,在p指针所指向的结点前插入一个指针q所指向的新结点,其修改指针的操作是( ) (假设双向循环链表的结点结构为(llink,data,rlink)。 A． p->llink=q； q->rlink=p；p->llink->rlink=q； q->llink=q；

B． p->llink=q； p->llink->rlink=q ； q->rlink= p；q->llink=p->llink；

C． q->rlink=p； q->llink=p->llink； p->llink->rlink=q; p->llink=q;

D． q->llink=p->llink；q->rlink=p； p->llink=q；p->llink=q；

**6.** 一棵完全二叉树上有1001个结点，其中叶子结点的个数是（ ）。

A． 250 B． 500 C．254 D．以上答案都不对

**7.** 已知一棵二叉树的前序遍历结果为ABCDEF, 中序遍历结果为CBAEDF, 则后序遍历的结果为（ ）。

A．CBEFDA B． FEDCBA C． CBEDFA D．不定

**8.** 利用二叉链表存储树时，则根结点的右指针是（ ）。

A．指向最左孩子 B．指向最右孩子 C．空 D．非空

**9.** 设有二维数组A[0..9, 0..19], 其中每个元素占两个字节，第一个元素的存储地址为100，若按列优先顺序存储，则元素A[6,6]存储地址为( )。

A. 252 B. 132 C. 352 D.232

10. 引入二叉线索树的目的是（ ）

A．加快查找结点的前驱或后继的速度

B．为了能在二叉树中方便的进行插入与删除

C．为了能方便的找到双亲

D．使二叉树的遍历结果唯一

1. **填空题：（每小题2分，共20分）**

1. 下面程序段中划线部分的执行次数为 。

int i=0, s=0;

while (++i<=n) {

int p=1;

for (int j=1;j<=i;j++) p\*=j;

s=s+p;

}

2. 向一个栈顶指针为H的链栈中插入一个s所指结点时, 执行的语句是 。

3. 如果一棵Huffman树T有n0个叶子结点， 那么树T中共有 个结点。

4. 在带有一个头结点的链队列front中，判定只有一个结点的条件是 。

5. 对于一个具有n个结点的单链表，在已知p所指向结点后插入一个新结点的时间复杂度是 ；在给定值为x的结点后插入一个新结点的时间复杂度是 。

6.一棵有n（n>0）个结点的满二叉树共有 个叶子和 非终端结点。

7. 有一个100\*90的稀疏矩阵（元素类型为整型），非0元素有10个，设每个整型数占2字节，则用三元组表示该矩阵时，所需的字节数是 。

8. 树的后根遍历序列等同于对该树对应的二叉树进行( )遍历的序列。

9．具有256个结点的完全二叉树的深度为\_\_\_\_\_\_。（假设根结点的深度为0）

10. 循环队列用数组P用（0,…,123）共n个元素表示，f为当前队列元素的前一位置，r为队尾元素的实际位置，当前队列f和r的值分别为100和32, 假定队列中元素个数总小于124, 则队列中元素个数为 。

1. **判断题：（每题1分，共10分）**

**1.**（ ）线性表若采用链式存储结构时，占用内存中存储单元的地址一定不连续。

**2.**（ ）线性表中每个元素都有一个前驱和一个后继。

**3.**（ ）广义表的长度就是广义表中的原子个数。

**4.**（ ）任意一棵二叉树中的结点的度都不大于2。

**5.**（ ）判断线索二叉树中由P所指结点是叶子结点的条件是(P->Lchild==NULL)&& (P->Rchild==NULL)。

**6.**（ ）采用三元组表方式对稀疏矩阵进行压缩存储时，三元组表中元素个数与矩阵中非零元素个数相同。

**7.**（ ）队列是一种运算受限的线性表。

**8.**（ ）二叉树的先序序列中的最后一个结点一定是叶子结点。

**9.**（ ）完全二叉树中，若一个结点没有左孩子，则它必是叶子结点。

**10.**（ ）两个栈共享一片连续内存空间时，为提高内存利用率，减少溢出机会，应把两个栈的栈底分别设在这片内存空间的两端。

**四、简答题: (每题5分，共20分)**

1**.** 设二叉树T的存储结构如下:

1 2 3 4 5 6 7 8 9 10

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Lchild | 0 | 0 | 2 | 3 | 7 | 5 | 8 | 0 | 10 | 1 |
| Data | J | H | F | D | B | A | C | E | G | I |
| Rchild | 0 | 0 | 0 | 9 | 4 | 0 | 0 | 0 | 0 | 0 |

其中BT为树根结点的指针，其值为Lchild,Rchild分别为结点的左、右孩子指针域,data为结点的数据域。

（l）画出二叉树T的逻辑结构；

（2）画出二叉树的后序线索树。

2. 用下面数据逐步建成堆。要求画出每加入一个关键码后堆的变化。

　　　（25　11　22　34　5　44　76　61　100　3 14 120）

**3**. 已知关键字集合 W={ 11, 8, 2, 3, 15, 9 }，以集合中的关键字作为叶子结点的权值而构造哈夫曼树（huffman Tree），画出构造的过程。

4. 设有一字符串P=”3\*y-a/y↑2”, 试写出利用栈将P改为”3y\*ay2↑/-”的操作步骤。（请用X代表扫描该字符串过程中顺序取一字符进栈的操作，用S代表从栈中取出一字符加入到新字符串尾的出栈操作。例如，要使”ABC”变为”BCA”，则操作步骤为XXSXSS）。

1. **算法设计题：（每题10分，共30分）**

1.已知L是带表头结点的单链表(表中元素个数 >= 2)，P指向某结点（非第一结点），删除P结点的直接前驱语句是：

2. 下面的算法是将整型数组A[0..n-1]中的元素划分为两部分，使得左边的所有元素均为奇数，右边的所有元素均为偶数，补充完成A,B,C,D四个空（每处空并非仅有一条语句）：

void Partition(int A[ ] )

{ i = 0; j = n-1;

while ( A )

{

while (i < j && B ) i++;

while (i < j && C ) j--;

if (i < j) D ;

}

}

3. 二叉树以二叉链表的方式存储，设计算法输出二叉树中所有的叶子结点，同时给出每个叶子结点到根结点的路径的长度。

一

1. C
2. B
3. D
4. B
5. C
6. D
7. A
8. C
9. D
10. A

二

1. n(n+1)/2
2. S->link = H; H= S;
3. 2n0-1
4. front->link->link==NULL; 或 front->link->link==rear;
5. O(1);O(n)
6. (n+1)/2;(n-1)/2
7. 60
8. 中序
9. 8
10. 56

三

1. F

2. F

3. F

4. T

5. F

6. T

7. T

8.T

9.T

10. T

四

1. 1) A

/

B

/ \

C D

/ / \

E F G

/ /

H I

/

J

2) 略

1. 3 5 14 34 11 22 76 61 100 25 44 120

3. 48

/ \

28 20

/ \ / \

15 13 9 11

/ \

5 8

/ \

1. 3

4.XSXXSSXXSXXSXXSSSS

五．

1. LinkNode \* tmp, \* pre;

tmp = L-> link; pre =L;

while (tmp->link != NULL && tmp->link != p)

{ pre = tmp; tmp = tmp -> link; }

if (tmp->link == NULL) return;

else

{ pre->link =p; delete tmp; }

1. A: i<j;

B: A[i] % 2

C: !(A[j] % 2)

D: A[i] +=A[j]; A[j]= A[i]- A[j]; A[i]= A[i]- A[j];

3

1．

解法1）

WPL(T:BNode \*):int;

{ n= 0; WPL1(T,0);

WPL= n

};

void WPL1(T:BNode \*; h:int);

{

if ( T !=NULL )

if ( (T->Lchild==NULL) && (T->Rchild==NULL)

) n= n+T->data\*h

; else { WPL1(T->Lchild,h+1); WPL1(T->Rchild,h+1) }

};

解法2）

WPL(T:BNode \*):int;

{

if ( T ==NULL

) WPL= 0

; else if ( (T->Lchild==NULL) && (T->Rchild==NULL)

) WPL= 0

; else WPL= T->data+WPL(T->Lchild+WPL(T->Rchild)

};