装订线 答题时不要超过此线

中 国 科 学 技 术 大 学

2022--2023 学年第一学期《数据结构》练习题

得分: _____ 专业:_____

姓名:_____ 学号:_____

一、单项选择题（2 分×10=20 分）

1. 下列程序段的时间复杂度为___A___。

   i=0，s=0； while (s<n) {s=s+i；i++；}

   A. $O(n^{1/2})$ B. $O(\log_2 n)$ C. O(n) D. $O(n^2)$

2. 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___B___。

   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；

3. 若元素 a、b、c、d、e、f 依次进栈，允许进栈、退栈操作交替进行，但不允许连续三次进行退栈操作，则不可能得到的出栈序列是___D___。

   A．d c e b f a B．c b d a e f C．b c a e f d D．a f e d c b

4. 设二维数组 A[1..m,1..n]按行存储在一维数组 B[1..mn]中，则二维数组元素 A[i,j]在一维数组 B 中的下标为___A___。

   A．n*(i-1)+j B． n*(i-1)+j-1 C．i*(j-1) D．j*m+i-1

5. 如果某二叉树中序序列的最后一个结点与后序序列的最后一个结点是同一个结点，则这棵二叉树____B____。

   A. 左子树为空 B. 右子树为空 C. 左、右子树都不空 D. 只有一个叶子

6. 5 个字符有如下 4 种编码方案，不是前缀编码的是___D___。

   A．01，0000，0001，001，1 B．011，000，001，010，1

   C．000，001，010，011，100 D．0，100，110，1110，1100

7. 中缀表达式(a-b)*(c+d)的后缀表达式是_____B_____。

   A. abcd+*- B. ab-cd+* C. ab-*cd+ D. a-bcd+*

8. 设 F 是一个森林，B 是由 F 转换得到的二叉树，F 中有 N 个非终端结点，则 B 中右指针域为空的结点有___C__个。

A N-1　　B N　　C N+1　　D N+2

9. 用单循环链表表示的队列,若仅设一个指针,但要求出队和入队操作方便,应当选用___B___。

A 头指针　　B 尾指针

10. 若仅已知某结点 A 的地址，则__A__结构不能删除 A 结点。

A 单链表　　B 双向链表　　C 循环链表

二、判断题。（正确的在括号里打 √，错误的打×，共 13 分）

1. 【×】线性表的长度是线性表所占用的存储空间的大小。
2. 【×】线性表的顺序存储结构优于链式存储结构。
3. 【×】顺序存储只能用于存储线性结构。
4. 【×】在对链队列做出队操作时，不会改变 front 指针的值。
5. 【×】算法的时间复杂度仅取决于问题的规模。
6. 【×】哈夫曼树中可能存在度为 1 的结点。
7. 【×】完全二叉树中，若一个结点没有右孩子，则必是树叶。
8. 【√】完全二叉树的某结点若无左孩子，则它必是叶结点。
9. 【×】若 A 和 B 都是一棵二叉树的叶子结点，则存在这样的二叉树，其前序遍历序列为…A…B…, 而中序遍历序列为…B…A…。
10. 【√】存在这样的二叉树，对它采用任何次序的遍历，其遍历产生的结点序列相同。
11. 【×】由一棵二叉树的前序序列和后序序列可以唯一确定它。
12. 【×】二叉树第 i 层上有 2^{i-1} 个结点，深度为 h 的二叉树上有 2^{h-1} 个结点。
13. 【√】将一棵树转换为二叉树后，根结点没有右子树。

三、填空题（1 分×13=13 分）

1. 二叉树第 i(根为的第一层)层上的结点数目最多为___2^{i-1}___；深度为 k 的

二叉树至多有__2^k-1___个结点。

2. 设高度为 h 的二叉树上只有度为 0 和度为 2 的结点，则此类二叉树中所含的结点数至少为_____2h-1_____，至多为_____2^h-1_____。
3. 用一维数组 Q[0..19]存放一个循环队列，队头指示器 front 指向队头元素的位置，队尾指示器 rear 指向队尾元素的下一个位置，队列中元素个数最多不超过 19 个。当 front＝18、rear＝3 时，该队列中有____5____个元素。若此后又进行了 4 次出队操作和 1 次入队操作，则 front 和 rear 的值分别为 2 和____4____。
4. 已知一棵二叉树的前序和中序序列分别为：ABCDEFG 和 CBDEAFG,它的后序序列是_____CEDBGFA_____。
5. 数据元素之间的关系在计算机中有两种不同的表示方法：顺序映像和非顺序映像，并由此得到两种不同的存储结构：顺序存储和链式存储。
6. 具有 n 个节点的完全二叉树的深度为$\lceil \log_2(n+1) \rceil$或 $\lfloor \log_2 n \rfloor+1$。
7. 若森林 F 有 15 条边、25 个结点，则 F 包含树的个数是____10_____。
8. 在长度为 n 的有序单链表中插入一个新结点，并仍然保持有序的时间复杂度是_____O(n)_____。
9. 包含 999 个结点的完全二叉树中的叶子结点数为___500___。

四、解答题（共 54 分）

1. 假设用于通信的电文仅由‘a’~‘h’的 8 个字母组成，字母在电文中出现的频率分别为 0.05,0.14,0.16,0.20,0.28,0.03,0.04,0.10。试给出这 8 个字母的哈夫曼编码方案；若电文由 100 个字母组成，请计算电文长度约有多少个二进制位。（15 分）

2. 假设一字符串用单循环链表表示，链表中每一结点存储一个字符，链表中最后一个结点存放的是字符串中的最后一个字符。试编写一算法判断该字符串是否有中心对称关系，例如，xyzzyx、xyzyx 都算是中心对称的字符串。（15 分）

```
int Judge(List*l){
    Stack S;
    int i;
    List*p = l;
    char ch;
    for(i = 0; i < l->length; i++){
        Push(S, p->data);
        p = p->next;
```

```
    }//P 中的元素入栈
    for(i = 0; i < l->length; i++, l=l->next){
        if(Pop(S,ch) && ch != l->data)
            break;
    }
    if(i == l->length)
        return True;
    else
        return False;
}
```

3. 假设二叉树中结点的存储结构定义如下：

typedef struct BTNode{

ElemType data;

struct BTNode *lchild, *rchild, *next;

}BTNode, *BiTree;

其中 lchild 和 rchild 用来保存结点的左、右孩子结点的指针，next 保存结点在后序遍历中的直接后继结点的指针。初始时，二叉树中各结点的 next 均为空。

(1) 试编写一算法，为二叉树中各结点的 next 填上合适的指针值，即其后序遍历的直接后继结点的指针。（8 分）

```
BTNode Pre;//全局变量记录上一个访问的结点
void FindNext(BTNode*T)
{
    if(T)
    {
        FindNext(T->lchild);
        FindNext(T->rchild);
        if(Pre)
        {
            Pre->next = T;
        }
        Pre = T;
    }
}
```

(2) 试编写一算法，给定两个指针 ptrx 和 ptry 指向树上节点，判断这两个节点是否是祖先和子孙关系。（8 分）

```
bool judge(BTNode * ptrx, BTNode *ptry){    // x 是 y 祖先 或 y 是 x 祖先，均返回 true
    if(!ptrx || !ptry) return false;
    bool func(BTNode * ptrx, BTNode *ptry);
    // ptrx == ptry 不算祖先/子孙关系
    return func(ptrx->lchild,ptry) || func(ptrx->rchild,ptry) || \
           func(ptry->lchild,ptrx) || func(ptry->rchild,ptrx);
}

bool func(BTNode * ptrx, BTNode *ptry){     // x 是 y 的祖先 或 x=y 时返回 true
    if(!ptrx) return false;
    if(ptrx == ptry) return true;
    return func(ptrx->lchild,ptry) || func(ptrx->rchild,ptry);
}
```

(3) 求出指定结点 ptr 在给定二叉树 root 中的层次。（假设根在第 1 层）（8 分）

```
int compute_deep(BTNode * root, BTNode* ptr){
    if(!root) return 0;
    if(ptr == root) return 1;
    int left = compute_deep(root->lchild, ptr), right = compute_deep(root->rchild,
ptr);
    if(left == right && left == 0) return 0;    // ptr 不在树 root 中
    return left > right ? left + 1 : right + 1; // left 与 right 有 1 个是 0，有一个不是 0
                                                //（ptr 不在左子树就是右子树）
}
```