**安徽大学20 20 —20 21 学年第 2 学期**

|  |
| --- |
| **院/系 年级 专业 姓名 学号**  **答 题 勿 超 装 订 线**  **------------------------------装---------------------------------------------订----------------------------------------线----------------------------------------** |
|  |
|  |

**《 数据结构 》考试试卷（A卷）**

**（闭卷 时间120分钟**）

**考场登记表序号**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **题 号** | **一** | **二** | **三** | **四** | **五** | **六** | **七** | **总分** |
| **得 分** |  |  |  |  |  |  |  |  |
| **阅卷人** |  |  |  |  |  |  |  |

|  |  |
| --- | --- |
| **得分** |  |

**一、算法分析题（每小题5分，共20分）**

1.分析下面算法的时间复杂度。

void Function(int n)

{ int i=1,j;

if (n ==1) return;

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

for (j=1; j <= n; j++) {

printf(“\*”);

break;

}

}

}

**2.**分析下面算法，回答问题。

LNode \*Function(LinkList L) //L为带头结点的单链表

{ LNode \*ptr1,\*ptr2;

int i=0;

ptr1=ptr2=L->next;

while (ptr1->next !=NULL) {

if (i == 0){

ptr1 = ptr1->next; i=1;

}

else if (i == 1){

ptr1 = ptr1->next; ptr2 = ptr2->next; i=0;

}

}

return ptr2;

}

(1) 请指出Function (L)算法的功能。

(2) 当L={1,3,5,7,9,11,13}时，执行Function(L)后，ptr2->data的值是多少？

**3.** 分析下面算法，回答问题。

int Function(BiTree \*root) // root 为二叉链表存储的二叉树

{ BiTNode \*temp; int n=0; Queue Q; //队列Q

if(!root) return 0;

InitQueue(Q); //InitQueue(&Q)为队列的初始化操作

EnQueue(Q,root); //EnQueue(&Q,e)为队列的入队操作

while(!IsEmptyQueue(Q)){

//IsEmptyQueue(Q)为队列的判空操作，若Q空则返回真，否则返回假

DeQueue(Q,temp);

if(!temp->lchild && temp->rchild || temp->lchild && !temp->rchild) n++;

if(temp->lchild) EnQueue(Q,temp->lchild);

if(temp->rchild) EnQueue(Q,temp->rchild);

}

DestroyQueue(Q); // DestroyQueue (&Q)为队列的销毁操作

return n;

}

(1) 请分析上述Function(root)算法的功能。

(2) 若root=(A(B(D,E),C(F,G)))，则执行Function(root)后，n等于多少？

**4.** 阅读并分析下面算法，回答问题。

KeyType Function(RecType R[],int s,int t,int k)

{ int i=s,j=t; RecType tmp;

if(s<t) {

tmp=R[s];

while(i<j){

while(j>i && R[j].key>=tmp.key) j--;

if(i<j) {R[i]=R[j];i++;}

while(i<j && R[i].key < tmp.key) i++;

if(i<j) { R[j]=R[i];j--}

}

R[i]=tmp;

if(k-1==i) return R[i].key;

else if (k-1<i) return Function(R,s,i-1,k);

else return Function(R,i+1,t,k);

}

else if(s==t && s==k-1) return R[k-1].key;

else return -1;

}

(1) 请分析上述Function(RecType R[],int s,int t,int k)算法的功能。

(2) 若R[0..9]序列中的关键字为{35,40,38,11,13,34,48,75,6,19}，执行Function (R,0,9,4)后,其结果为多少（函数的返回值）？

**二、计算题（每小题5分，共10分）**

|  |  |
| --- | --- |
| **得分** |  |

**5.** 已知广义表L=((a,b),(c,d))，请计算Tail(Head(Tail(L)))的运算结果。

|  |
| --- |
| **院/系 年级 专业 姓名 学号**  **答 题 勿 超 装 订 线**  **------------------------------装---------------------------------------------订----------------------------------------线----------------------------------------** |
|  |
|  |
|  |

**6.** 有三维数组a[0..7,0..8,0..9]采用按行序优先存储，数组的起始地址是1000，每个元素占用4个字节，请计算元素a[2,5,6]的起始地址。

**三、应用题（每小题10分，共40分）**

|  |  |
| --- | --- |
| **得分** |  |

**7.**假设一棵二叉排序树的先序序列为EBADCFHGIKJ，中序遍历序列为ABCDEFGHIJK，请画出该二叉树。

**8.**已知某一组记录的关键字序列R为(45,62,35,55,60,50,77,58,15,35\*,98)，请给出每一趟快速排序结束后关键字序列的状态。

**9.** 设散列函数H(Key)=Key % 7，散列地址空间为0-9，对关键字序列(8,38,20,12,27,

23),采用线性探测法处理冲突。构造散列表。试回答下列问题：

(1) 画出散列表示意图。

(2) 分别计算等概率情况下，查找成功和查找不成功时的平均查找长度。

**10.**已知某无向图，如下图所示，试用Prim算法，从顶点1出发，求其最小生成树。

2

3

6

7

5

4

1

3

5

4

**四、算法设计题（每小题10分，共30分）**

|  |  |
| --- | --- |
| **得分** |  |

**11.**给定两个递增的有序链表List1={A1,A2,…,An}和List2={B1,B2,…,Bm},请设计一个算法合并它们得到一个新链表，新链表形如：

如果n>=m，那么新链表为{A1,B1,A2,B2,…,Am,Bm,Am+1,…,An}；

如果n<=m，那么新链表为{A1,B1,A2,B2,…,An,Bn,Bn+1,…,Bm}。

|  |
| --- |
| **院/系 年级 专业 姓名 学号**  **答 题 勿 超 装 订 线**  **------------------------------装---------------------------------------------订----------------------------------------线----------------------------------------** |
|  |
|  |
|  |

**12.** 二叉树采用二叉链表存储，其类型定义如下：

typedef struct BiTNode{

TElemType data;

struct BiTNode \*lchild,\*rchild;

}BiTNode,\*BiTree;

请设计下列算法，(1)删除一棵二叉树；(2)求二叉树的高度（或深度）。

**13.** 假设以带头结点的循环链表表示队列，并且只设一个指针指向队尾元素结点（注意：不设头指针），其类型定义如下：

typedef struct QNode{

QElemType data;

struct QNode \*next;

}QNode,\*QueuePtr;

typedef struct {

QueuePtr rear;

}LinkQueue;

请设计队列的以下算法：

1. 初始化操作 /\* void InitQueue(LinkQueue &Q) \*/
2. 入队操作 /\* void EnQueue(LinkQueue &Q, QElemType e) \*/
3. 出队操作 /\* void DeQueue(LinkQueue &Q, QElemType &e) \*/