**算法题参考答案**

本题无统一答案。每道题编写算法时完成题目功能即可。

1．设顺序表L中的数据元素递减有序，编写一个算法，将数据元素x插入到顺序表L的适当位置上，以保持该顺序表的有序性。

顺序表的类型描述：

#define    maxsize 线性表可能达到的最大长度；

typedef  struct

{ ElemType  elem[maxsize]；  /\* 线性表占用的数组空间。\*/

  int        last；    /\*记录线性表中最后一个元素在数组elem[ ]中的位置（下标值），空表置为-1\*/

} SeqList；

参考答案：

void SLOrderInsert (SeqList \*L，ElemType x)

{ int i；

if(L->last>=MaxSize-1)

{ printf(“\nOverflow!\n”)；

return；

}

/\*顺序表L中从下标 L.last至i的数据元素依次后移一个位置\*/

for(i=L->last；i>=0 && x<L->elem[i]；i--)

L->elem[i+1]=L->elem[i]；

L->elem[i+1]=x； /\*把数据元素x插入\*/

L->last++；

}

评分标准：

参数正确：2分

移动正确：6分

插入正确及长度修改正确：2分

方法二：

int Linsert(SeqList \*L,int X)

{ int i=0,k;

if(L->last>=MAXSIZE-1) /\*如果表满，无法插入X，返回0\*/

{ printf(“表已满无法插入”)；

return(0);

}

while(i<=L->last&&L->elem[i]<X) /\*从第一个元素开始比较，找到待插入位置i\*/

i++;

for(k=L->last;k>=i;k--) /\*从最后一个元素开始将第n～i个元素后移一位\*/

L->elem[k+1]=L->elem[k];

L->elem[i]=X; /\*将X插入到第i个位置上\*/

L->last++; /\*表长加1\*/

return(1);

}

评分标准：

参数定义正确：2分

合法性判断：2分

正确移动元素：4分

正确插入元素：1分

正确修改表长：1分

2.二叉树采用二叉链表形式存放，编写递归算法，采用扩展先序遍历序列创建一棵二叉树的二叉链表。

参考答案：

void CreateBiTree(BiTree \*bt)

{ char ch;

  ch=getchar();

  if(ch=='.') \*bt=NULL;

  else

{

\*bt=(BiTree)malloc(sizeof(BiTNode));

    (\*bt)->data=ch;

    CreateBiTree(&((\*bt)->LChild));

    CreateBiTree(&((\*bt)->RChild));

   }

}

评分标准：

参数正确：2分

条件正确：2分

正确建立二叉链表：6分

3. 设单链表head中的数据元素递增有序，编写一个算法，将数据元素x插入到单链表中的适当位置上，以保持该链表的有序性。

int Linsert(LinkList head,int x)

{ Node \*p,\*s;

p=head;

while(p->next!=NULL&&p->next->data<x)

p=p->next;

s=(Node \*)malloc(sizeof(Node));

s->data=x;

s->next=p->next;

p->next=s;

}

评分标准：

参数定义正确：2分

找到插入位置：4分

正确插入新结点：4分

4．已知一个二叉树采用二叉链表存放，写一算法，计算二叉树的深度。

int dep=0;

void Depth(BiTree t，int level)

{if(t!=NULL)

{if(level>dep) depth=level;

Depth(t->LChild, level+1);

Depth(t->RChild, level+1);

}

}

评分标准：

参数定义正确：2分

变量定义正确：1分

条件正确：3分

正确递归调用：4分

5. 假设以带头结点的单链表表示非递减有序表，设计一算法删除表中所有值大于min且小于max（假设min<max）同时释放结点空间。（共10分）

int Delete\_Between(LinkList L, int max, int min)

{p=L;

while(p->next->data<=min) p=p->next;

if(p->next)

{q=p->next;

while(q->data<max) {k=q; q=q->next; free(k); }

p->next=q;

}

}

评分标准：循环指针p赋初值2分，找到要删除第一个结点3分，删除结点修改链5分。

6. 编写在二叉排序树中查找关键字K的算法。（共10分）

BSTree SearchBST（BSTree bst ，KeyType key）

{if(!bst) return NULL;

else if（bst->key==key）return bst;

else if (bst->key>key)

return SearchBST(bst->lchild,key);

else

return SearchBST(bst->rchild,key);

}

评分标准：查找条件正确2分，递归查找左子树4分，递归查找右子树4分。

7．假设有一个带头结点的循环链表L的长度大于1，试编写算法在此链表中删除尾结点。

void Dele(LinkList L)

{ Node \*p,\*r;

p=L;

while(p->next->next!=L)

p=p->next;

r=p->next;

p->next=r->next;

free(r);

}

评分标准：

参数定义正确：2分

正确找到尾结点的前驱：4分

正确删除尾结点：4分

8．已知一个二叉树采用二叉链表存放，写一算法，要求统计出二叉树中度为2的结点个数。

int count=0;

Void Degree1(BiTree t)

{if(t!=NULL)

{if(t->lchild&&t->rchild)

count++;

Degree1(t->lchid);

Degree1(t->rchid);

}

}

评分标准：

参数定义正确：2分

变量定义正确：1分

条件正确：3分

正确递归调用：4分

**9.** 假设在长度大于1的循环链表中，既无头结点也无头指针。s为指向链表中某个结点的指针，编写算法删除指针s所指结点的前趋结点。

void Dele(Node \* s)

{ Node \*p,\*r;

p=s;

while(p->next->next!=s)

p=p->next;

r=p->next;

p->next=r->next;

free(r);

}

评分标准：

参数定义正确：2分

正确找到s的前驱：4分

正确删除结点：4分

10. 设计一个在二叉链表存储结构上统计二叉树中结点个数的算法。

int count=0；

void countnode(bitree bt)

{

if(bt!=NULL)

{count++;

countnode(bt->LChild);

countnode(bt->RChild);}

}

评分标准：

参数定义正确：2分

变量定义正确：1分

条件正确：3分

正确递归调用：4分

11. 设头指针为head，编写算法实现带头结点的单链表head的就地逆置，即利用原带头结点的单链表 head的结点空间把数据元素序列(a1，a2，……an)逆置为(an，an-1，……，a1)

参考答案：

void ConverseLink (LinkList head)

{ Node \*p**，** \*q**；**

p=head->next； /\*p为不带头结点的单链表的头指针 ，指向剩下链表中第一个结点\*/

head->next=NULL /\* 使head指向带头结点的空单链表 \*/

while(p!=NULL) /\* 循环从链表头部插入 \*/

{q=p； /\*q指向正在处理的结点 \*/

p=p->next； /\*p指向原结点的下一个结点 \*/

q->next=head->next； /\*把q所指结点链入链表 \*/

head->next=q； /\*q所指结点为链表的首结点 \*/

}

}

评分标准：

参数正确：2分

链表头指针及头结点设置正确：2分

各结点逆置正确：6分

12．已知一个二叉树采用二叉链表存放，编写算法，在二叉树中求位于先序序列中第k个位置的结点的值。

参考答案：

void CountNum(BiTree root, int k,int i)

{

  if(root && i!=k)

  {

i++;

if(i==k) printf("%c",root->data);

else

  {

     CountNum(t->LChild,k,i);

 CountNum(t->RChild,k,i);

   }

}

}

评分标准：

参数正确：2分

条件正确：2分

正确求出第k个值：6分

13．设计在单链表中删除值相同的多余结点的算法。

typedef int datatype;

typedef struct node

{ datatype data;

  struct node \*next;

}lklist; （2分）

void delredundant(lklist \*head)

{

  lklist \*p,\*q,\*s;

  for(p=head->next;p!=NULL;p=p->next)

  {

    for(q=p->next,s=q;q!=NULL; ) （3分）

      if (q->data==p->data)

       {p->next=q->next;

q=q->next;

        free(s);         （2分）

        s=q;}

      else

        {s=q；

q=q->next; （3分）

        }

    }

}

评分标准：数据类型定义2分；写出查找相同结点循环语句3分；找到要删除结点修改链2分，找不到修改链3分。

14．已知一个二叉树采用二叉链表存放，写一算法，要求统计出二叉树中度为1的结点个数。

typedef struct node

{ datatype data;

 struct node \*lchild,\*rchild;

} bitree;

void fun(bitree \*root) （2分）

{

  if (bt!=NULL)

  {if(root->lchild!=NULL&&root->rchild==NULL)

Count++;

else if(root->lchild==NULL&&root->rchild!=NULL) （4分）

Count++;

}

fun(root->lchild); （2分）

fun(root->rchild); （2分）

}

评分标准：写出数据结构类型2分；写出度为1结点判定条件4分；递归操作左右子树各2分。

15. 设有一个由正整数组成的无序单链表，编写完成下列功能的算法：

（1）找出最大值结点，且打印该数值；

（2）若该数值是偶数，则将其与直接后继结点的数值交换；

参考答案：

void fun(LinkList head)

{

int max,t;

Node \*p**, \***q;

p=head->next; q=p;

if(p!=NULL)

{ max=p->data;

while(p!=NULL)

{

if(max<p->data) {max=p->data;q=p;}

p=p->next;

}

printf(“max=%d\n”,max);

if(max%2==0 && q->next!=NULL)

{

t=q->data;

q->data=q->next->data;

q->next->data=t;

}

}

评分标准：

参数正确：2分

找到最大值位置并输出：6分

正确互换2分

16．已知一个二叉树采用二叉链表存放，设计一个算法将所有结点的左、右子树相互交换。

参考答案：

void convert (BiTree root)

{

BiTNode \*temp;

if(root!=NULL)

{

if(root->lchild!=NULL ||root->rchild!=NULL)

{

temp=root->lchild;

root->lchild=root->rchild;

root->rchild=temp;

}

convert(root->rchild);

convert(root->lchild);

}

}

评分标准：

参数正确：2分

正确交换：8分

17．设有一个由正整数组成的无序单链表，编写完成下列功能的算法：

（1）找出最小值结点，且打印该数值；

（2）若该数值是偶数，则将其与直接后继结点的数值交换；

单链表类型描述：

typedef struct Node

{ ElemType data；

  struct Node  \* next；

}Node, \*LinkList；

参考答案：

void fun(LinkList head)

{

int min;

Node \*p**, \***q;

p=head->next; q=p;

if(p!=NULL)

{

min=p->data;

while(p!=NULL)

{

if(min>p->data) {min=p->data; q=p;}

p=p->next;

}

printf(“min=%d\n”,min);

if(min%2==0 && q->next!=NULL)

{

p=q->next;

q->next=p->next;

free(p);

}

}

评分标准：

参数正确：2分

找到最小值位置并输出：6分

正确删除2分

18已知一个二叉树采用二叉链表存放，写一算法，统计出二叉树中叶子结点的个数。

参考答案：

Int node=0; /\*node为保存叶子结点数目的全局变量，调用之前初始化为0 \*/

void CountNode(BinTree root) /\*求二叉树中叶子结点个数\*/

{

if (root!=NULL)

{ CountNode(root->lchild); /\* 中序遍历左子树\*／

if ((root->lchild==NULL)&&(root->rchild==NULL))

node++; /\*结点计数\*／

CountNode(root->rchild); /\* 中序遍历右子树\*／

}/\*if\*/

}/\* CountNode \*/

评分标准：

参数正确：1分

计数器定义正确：2分

正确计数：7分

19.在带头结点的单链表head的结点a之后插入新元素x。 （共10分）

typedef struct Node

{Elemtype data;

Struct Node \*next;

}Node,\*LinkList;

Void InsertList(LinkList L,Elemtype x,Node \*a)

{LinkList p=L->next;

while(p!=null&&p!=a)p=p->next;

if(p)

{s=(LinkList)malloc(sizeof(Node));

s->data=x;

s->next=p->next;

p->next=s;

}

}

评分标准：写出数据类型定义2分；找到插入结点a3分；生成新结点s2分；修改链完成插入3分。

20. 二叉树采用链接存储结构，试设计一个按层次顺序（同一层次自左至右）遍历二叉树的算法。（共10分）

typedef char datatype;

typedef struct node

{datatype data;

struct node \*lchild,\*rchild;

} bitree;

int LayTraverse(bitree \*bt)

{InitQueue( Q);

p=bt;

if(bt==null)return error;

EnterQueue(Q,bt);

while(!IsEmpty(Q))

{DeleteQueue(Q,p);

visit(p);

if(p->lchild) EnterQueue(Q, p->lchild);

f(p->rchild) EnterQueue(Q, p->rlchild);

}

return ok;

}

}

评分标准：写出数据类型定义2分；初始化队列，根指针进队2分；结点的邻接点依次进队，进行循环6分。