2001年试题参考答案

数据结构实题

（一）

先序遍历：A B D C E F

中序遍历：D B A E C F

后序遍历：D B E F C A

（二）

栈	顺序栈	链栈

初始化	void initStack(Sqstack &st)
	{
	st.top=-1;
	}
		void initStack(LNode *lst)
		{
		lst=(LNode*)malloc(sizeof(LNode));
		lst->next=NULL;
		}

进	int push(Sqstack &st,int x)
	{
	if(st.top==maxsize-1)
	return 0;
	st.data[++(st.top)]=x;
	return 1;
	}
		void push(LNode *&lst,int x)
		{
		LNode *p;
		p=(LNode*)malloc(sizeof(LNode));
		p->next=NULL;
		p->data=x;
		p->next=lst->next;
		lst->next=p;
		}

出	int push(Sqstack &st,int &x)
	{
	if(st.top==-1)
	return 0;
	x=st.data[(st.top)--];
	return 1;
	}
		int pop(LNode *&lst,int &x)
		{
		LNode *p;
		if(lst->next==NULL)
		return 0;
		p=lst->next;
		x=p->data;
		lst->next=p->next;
		free(p);
		return 1;
		}

	顺序队	链队

初始化	void initQueue(SqQueue &qu)
	{
	qu.front=qu.rear=0;
	}
		void initQueue(LiQueue *&lqu)
		{
		lqu=
		(LiQueue*)malloc(sizeof(LiQueue));
		lqu->front= NULL;
		lqu->rear=NULL;
		}

进	int enQueue(SqQueue &qu,int x)
	{
	if((qu.rear+1)%maxsize==qu.front)
	return 0;
	else
	{
	qu.rear=(qu.rear+1)%maxsize;
	qu.data[qu.rear]=x;
	return 1;
	}
	}
		void enQueue(LiQueue *&lqu,int x)
		{
		QNode *p;
		p=
		(QNode*)malloc(sizeof(QNode));
		p->data=x;
		p->next=NULL;
		if(lqu->rear==NULL)
		lqu->front=
		lqu->rear=p;
		else

		{
		lqu->rear->next=p;
		lqu->rear=p;
		}
		}

出	int deQueue(SqQueue &qu,int &x)
	{
	if(qu.front==qu.rear)
	return 0;
	else
	{
	qu.front=(qu.front+1)%maxsize;
	x=qu.data[qu.front];
	return 1;
	}
	}
		int deQueue(LiQueue *&lqu,int &x)
		{
		QNode *p;
		if(lqu->rear==NULL)
		return 0;
		else
		p=lqu->front;
		if(lqu->front==lqu->rear)
		lqu->front=
		lqu->rear=NULL;
		else
		lqu->front=
		lqu->front->next;
		x=p->data;
		free(p);
		return 1;
		}

（三）参照课本或者基础篇例题、习题。

（四）

双向链表结点结构：

```
typedef struct DLNode
{
        ElemType data;
        struct DLNode *pre;
        struct DLNode *nex;
}DLNode;
```

删除子程序：

```
p=s->pre;
p->nex=s->nex;
s->nex->pre=p;
free(s);
```

（五）

```
int count(AGraph g,int k)
{
    ArcNode *p;
    int i,sum=0;
    for(i=1;i<g.vexnum;i++)
    {
            p=g.adjlist[i].first;
            while(p)
```

```
        {
                if(p->adjvex==k)
                {
                        sum++;
                        break;
                }
                p=p->next;
        }
}
return sum;
}
void main()
{
    AGraph g;
    int i;
    for(i=1;i<g.vexnum;i++)
    {
            cout<<count(g,i)<<" ";
    }
}
```

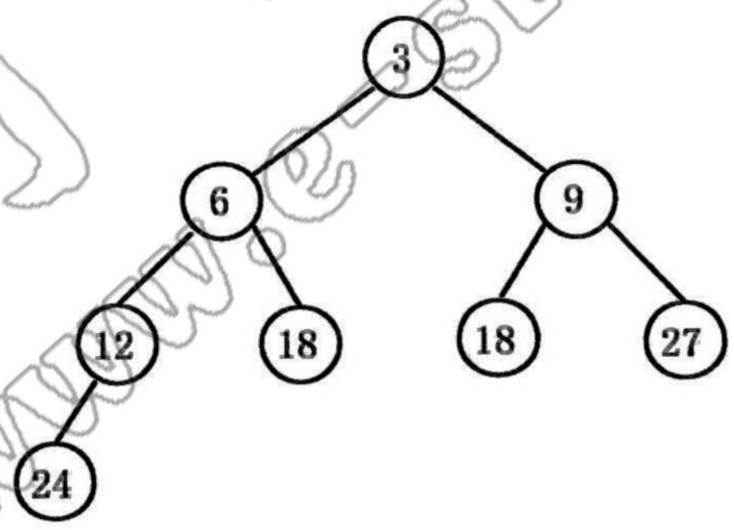
（六）程序阅读题

1、

运行结果：

3 6 12 24 18 9 18 27

结果分析：



本题的最佳解法就是运用树的形式，首先输出 3 之后执行 exa(6)、exa(9)，这里需要注意需要采用回溯的策略，在 3 的左子树执行完毕后才能开始输出右子树。可以读出本题其实是此二叉树的先序遍历序列：3,6,12,24,18,9,18,27。

2、

运行结果：

0,2,1

3、

运行结果：

3 2 1