



1. 试将下列递归过程改写为非递归过程。
 void test(int &sum) {
 int x; cin>>x;
 if(x==0) sum=0;
 else {
 test(sum);
 sum+=x;
 }
 cout<<sum;
}

答：

```

void test(int &sum){
    Stack s; InitStack(s); int x;
    do{   cin>>x;   Push(s,x); }while(x!=0);
    while(!StackEmpty(s)){ Pop(s,x);  sum+=x;  cout<<sum<<endl; }
    DestoryStack(s);
}

```

思路类似于线性递归，但需要用到栈。

这里关键是有个“cout<<sum”，否则就可以用一个简单的for循环求累加和。

- 2. 设有广义表D(a,b,D),其长度为 (), 深度为 ()
 - A ∞ B 3 C 2 D 5
- 3. 利用广义表的Head和Tail操作写出函数表达式,把原子banana分别从下列广义表中分离出来
 - (1) $L_1 = (((apple)), ((pear)), (banana), orange);$
 - (2) $L_2 = (apple, (pear, (banana), orange));$

答:

1. B, A

2.

(1) $Head(Head(Tail(Tail(L_1))));$

(2) $Head(Head(Tail(Head(Tail(L_2)))))$;

写一个判断两个广义表相等的递归算法

- 方法的声明如下：
- template <class Type> bool GenList
<Type> ::Equal(GenListNode <Type> *s,
GenListNode <Type> *t)
- 第一次调用时，其中的s和t是2个广义表的头结点

分析：

判断两个广义表是否相等，不但两个广义表具有相同的结构，而且对应的数据成员具有相等的值。分为以下3种情况：

如果两个广义表都是空表，则相等。

如果两个广义表的对应结点都是原子结点，对应项的值相等，再递归比较同一层后面的表元素。

如果两个广义表中对应项是子表结点，则递归比较相应的子表。

```
template <class Type> bool GenList <Type> ::Equal(GenListNode <Type> *s,  
GenListNode <Type> *t) {  
    int x;  
    if (s->tlink==NULL && t->tlink==NULL) return true; //表s与表t都是空表或者所有  
    结点都比较完  
    if (s->tlink!=NULL && t->tlink!=NULL && s->tlink->utype==t->tlink->utype) //两表  
    都非空且结点标志相同  
    {  
        if (s->tlink->utype==1) x=(s->tlink->info.value==t->tlink->info.value) ? 1 : 0; //  
        原子结点，比较对应数据  
        else if (s->tlink->utype==2) x=Equal(s->tlink->info.hlink, t->tlink->info.hlink);  
        if (x==1) return Equal(s->tlink, t->tlink);  
    }
```

```
}  
return false;  
}
```