第3章习题

- 3.3 写出下列程序段的输出结果(栈的元素类型SElemType为char)

```
void main(){
   Stack S;
   char x,y;
   InitStack(S);
   x='c'; y='k';
   Push(S,x); Push(S,'a');  Push(S,y);
   Pop(S,x);  Push(S,'t');  Push(S,x);
   Pop(S,x);  Push(S,'s');
   while(!StackEmpty(S)) {Pop(S,y); printf(y);};
   printf(x);   }
```


- 3.4 简述以下算法的功能(栈的元素类型SElemType为int)

```
(1)
status algo1(Stack S){
   int i, n, A[255];
   n=0;
   while(!StackEmpty(S))
   {   n++;
       Pop(S, A[n]);    };
   for(i=1; i<=n; i++)
      Push(S, A[i]);
}
```

```
(2)
status algo2(Stack S, int e){
   Stack T; int d;
   InitStack(T);
   int i, n, A[255];
   n=0;
   while(!StackEmpty(S)) {
      Pop(S, d);
      if(d!=e) Push(T,d);  }
   while(!StackEmpty(T)) {
      Pop(T, d);
      Push(S,d);   }
}
```


- 3.13 简述以下算法的功能(栈和队列的元素类型均为int)

```
void algo3(Queue &Q){
   Stack S; int d;
   InitStack(S);
   while(!QueueEmpty(Q)) {
      DeQueue(Q, d);
      Push(S,d);  }
   while(!StackEmpty(S)) {
      Pop(S, d);
      EnQueue(Q,d);   }
}
```


- 3.15 假设以顺序存储结构实现一个双向栈，即在一维数组的存储空间中存在着两个栈，它们的栈底分别设在数组的两个端点。试编写实现这个双向栈tws的三个操作：初始化inistack(tws)、入栈push(tws,i,x)和出栈pop(tws,i,x)的算法，其中i为0或1，用以分别指示设在数组两端的两个栈。

- 3.17 试写一个算法，识别依次读入的一个以@为结束符的字符序列是否为形如'序列1&序列2'模式的字符序列。其中序列1和序列2中都不含字符'&'，且序列2是序列1的逆序列。例如，'a+b&b+a'是属该模式的字符序列，而'1+3&3-1'则不是。
- 3.25 试写出递归函数F(n)的递归算法，并消除递归：

$$F(n)=\begin{cases} n+1 & n=0 \\ n\bullet F(n/2) & n>0 \end{cases}$$

- 3.29 如果希望循环队列中的元素都能得到利用，则需设置一个标志域tag，并以tag的值为0或1来区分，尾指针和头指针值相同时的队列状态是"空"还是"满"。试编写与此结构相应的入队列和出队列的算法，并从时间和空间角度讨论设标志和不设标志这两种方法的使用范围(如当循环队列容量较小而队列中每个元素占空间较多时，哪一种方法较好)。