栈、队列与链表

请注意：这是第一版材料，不排除出现错误，如果觉得有问题请及时提出

栈

栈是一种非常简单的数据结构，其基本逻辑就像一个桶，向桶里放东西叫做入栈，从桶的最上面把东西拿走叫做出栈，下面是对栈的代码的演示

1.栈的创建

```
s=[0]*4
top=-1
```

2.入栈

```
top+=1
s[top]='要插入的元素'
```

3.出栈

```
print(s[top])
top-=1
```

4.栈满

```
if top+1==len(s):
```

例栈应用实例

> 这是一道判断式子括号个数是否成对的程序（括号嵌套不得超过10个）
> 基本思路：
> 碰到左括号就入栈，碰到右括号就出栈，如果到最后还有剩下的左括号或者出栈时栈为空则说明括号不成对

```
s=[0]*10
top=-1
flag=True
a=list(input())
for i in a:
    if i=='(':
        top+=1
        s[top]=1
    elif i==')':
        if top>=-1:
            top-=1
        else:
            flag=False
            break
if flag and top==-1:
    print('式子正确')
else:
    print('式子错误')
```

队列

队列分为两种：普通队列和循环队列

普通队列

普通队列是一种十分简单的数据结构，其基本逻辑就像一个管道，进入管道的一头叫队尾，出管道的一头叫做队首，进入管道就叫入队，出管道就叫出队，下面是栈的代码演示：

1.队列的创建

```
que=['']*8
head=0
tail=0
```

2.普通队列-入队

```
que[tail]='要入队的内容'
tail=tail+1
```

3.普通队列-出队

```
print(que[head],end='')
head=head+1
```

4.普通队列-判断队为满

```
if tail==len(que):
```

5.普通队列-判断队为空

```
if tail==head:
```

6.普通队列-长度判断

```
lengq=tail-head
```

循环队列

循环队列就是把普通队列的笔直管道弯成一个圆形，让空间可以多次利用，更省内存，一下是循环队列的代码演示

1.队列的创建

```
que=['']*8
head=0
tail=0
```

2.循环队列-入队

```
que[tail]='要入队的内容'
tail=(tail+1)%len(que)
```

3.循环队列-出队

```
print(que[head],end='')
head=(head+1)%len(que)
```

4.循环队列-判断队为满

```
if  (tail+1)%len(q)==head:
```

5.循环队列-判断队为空

```
if tail==head:
```

6.循环队列-长度判断

```
lengq=(tail+len(q)-head)%len(q)
```

队列应用实例

> 这是一道加密题：
> 给定一个字符串,按如下过程加密：取出第一个字符S1，将第二个字符S2，放到字符串的末尾Sn后面，得到信字符串；接着把S3取出，S4放到字符串的末尾S2，后面……直到最后一个字母Sn被取出。这些字母按取出的顺序形成一个新的字符串，称为密串。下面的程序实现了输入一个字符串（长度小于等于10），输出该字符串的密串
> 基本思路：
> 先全部入队，然后每三个取出一个，否则直接从队首加到队尾

```
s=input('请输入一个字符串')
que=['']*10
head=0
tail=0
print('秘串是：', end='')
for i in range(len(s)):
    que[tail]=s[i]
    tail=(tail+1)%len(que)
while tail!=head:
    print(que[head],end='')
    head=(head+1)%len(que)
    if tail!=head:
        que[tail]=que[head]
        tail=(tail+1)%len(que)
        head=(head+1)%len(que)
```

链表

链表是一种复杂的数据结构，一下是栈的代码演示：

1.链表创建

```
data=[['E',-1],['D',0],['B',4],['A',2],['C',1]]
head=3
```

2.遍历

```
data=[['E',-1],['D',0],['B',4],['A',2],['C',1]]
head=3
p=head
while p!=-1:
    print(data[p][0],end='->')
    p=data[p][1]
```

(之后的代码用自定义函数 `read(data)` 来表示遍历)

3.向元素之后插入

```
data=[['D',-1],['B',3],['A',1],['C',0]]
head=2
```

```
cp=head
while cp!=-1:
    if data[cp][0]=='B':
        data.append(['BB',data[cp][1]])
        data[cp][1]=len(data)-1
        break
    cp=data[cp][1]
read(data)
```

4.头节点插入

```
data=[['D',-1],['B',3],['A',1],['C',0]]
head=2
cp=head     #指针
pp=head     #前驱指针（储存指针的上一个值）
while cp!=-1:
    if data[cp][0]=='C':    #图中if条件指此条件
        data.append(['BB',cp])
        data[pp][1]=len(data)-1
        break
    pp=cp
    cp=data[cp][1]
read(data)
```

5.删除非头节点

```
data=[['B',2],['D',-1],['C',1],['A',0]]
head=3
cp=head
pp=head
while cp!=-1:
    if data[cp][0]=='B':
        data[pp][1]=data[cp][1]
        break
    pp=cp
    cp=data[cp][1]
read(data)
```

6.删除头节点

```
data=[['B',2],['D',-1],['C',1],['A',0]]
head=3
head=data[head][1]
read(data)
```

链表应用例题

> 大家都很熟悉的两个升序链表合成一个升序链表

```
def read(data,head):
    tmp=head                                    #这里tmp代替指针
    while tmp!=-1:
        print(a[tmp][0],end=" ")
        tmp=a[tmp][1]
    print()
a=[[1,1],[3,2],[4,3],[8,4],[16,-1]]             #链表A
b=[[2,1],[5,2],[7,3],[10,4],[15,-1]]            #链表B
heada=headb=0                                   #头节点都为0
ka=kb=qa=0
while kb!=-1 and ka!=-1:                        #当没遍历完时
    if a[ka][0]>b[kb][0]:                       #判断是否需要插入（如果a的当前项>b的当前项，就要在a的
当前项前插入b的当前项）
        if ka==heada:                           #判断是否为头节点，如果是就是头节点插入，不是就是普通节
点插入
```

```
            a.append([b[kb][0],heada])
            heada=len(a)-1
        else:                                   #普通节点插入
            a.append([b[kb][0],ka])
            a[qa][1]=len(a)-1
        qa=len(a)-1                             #插入节点之后把a的前驱指针指向
        kb=b[kb][1]                             #b的指针下移
    else:                                       #否则就a的指针下移
        qa=ka
        ka=a[ka][1]
while kb!=-1:
    a.append([b[kb][0],ka])                     #把b中剩余的全部加入a
    a[qa][1]=len(a)-1
    qa=len(a)-1                                 #也可以写成qa=a[qa][1]
    kb=b[kb][1]
read(a,heada)                                   #遍历
```