9.27

1.选择题

（1）C

> C选项中1，2的顺序不能实现。

（4）A

> 应先将结点的值保存到x中，再将栈顶结点的指针指向下一结点，即可进行删除栈顶结点的操作。

2.算法设计题

（5）

①AD均为合法

> 对于B，进栈一次，出栈一次后栈为空，无法再进行出栈操作。
> 对于C，一共进栈五次，出栈三次，导致栈不为空，所以是非法操作。

②判断操作序列是否合法

```
bool Judge(char s[]){
    int i=0,j=0,k=0;//i表示第i个字符，j表示入栈个数，k表示出栈个数
    while(s[i]!='\0'){
        switch(s[i]){//每进行一步，则进行计数操作
            case 'I':j++;
                    break;
            case 'O':k++;
                    break;
            default:return false;
        }
        if(j<k) return false;//入栈数小于出栈数则非法
        i++;
    }
    if(j!=k) return false;//栈不为空也非法
    return true;
}
```

（9）

①Ack(m,n)递归算法

```
int Ack1(int m,int n){
    int t;
    if(m==0) return n+1;
    else if(n==0) return Ack1(m-1,1);
    else{
        t=Ack1(m,n-1);
        return Ack1(m-1,t);
    }
}


Ack(2,1)=Ack(1,Ack(2,0))
        =Ack(1,Ack(1,1))
        =Ack(1,Ack(0,Ack(1,0)))
        =Ack(1,Ack(0,Ack(0,1)))
        =Ack(1,Ack(0,2))
        =Ack(1,3)
        =Ack(0,Ack(1,2))
        =Ack(0,Ack(0,Ack(1,1)))
        =Ack(0,Ack(0,Ack(0,Ack(1,0))))
        =Ack(0,Ack(0,Ack(0,Ack(0,1))))
        =Ack(0,Ack(0,Ack(0,2)))
        =Ack(0,Ack(0,3))
        =Ack(0,4)
        =5
```

②Ack(m，n)非递归算法

```
int Ack2(int m,int n){
    struct{
        int am,an;  //分别保存m和n值
        int af;     //保存akm(m,n)值
        int flag;  //标识是否求出akm(m,n)值，1:表示未求出，0:表示已求出
    }St[MaxSize];

    int top=0;  //栈指针
    top++;     //初值进栈
    St[top].am=m; St[top].an=n; St[top].flag=1;
    while(top>0){   //栈不空时循环
        if (St[top].flag==1)   //未计算出栈顶元素的af值
        {
            if (St[top].am==0)      //(1)式
            {
                St[top].af=St[top].an+1;
                St[top].flag=0;
            }
            else if (St[top].an==0) //(2)式
            {
```

```
                top++;
                St[top].am=St[top-1].am-1;
                St[top].an=1;
                St[top].flag=1;
            }
            else                    //(3)式
            {
                top++;
                St[top].am=St[top-1].am;
                St[top].an=St[top-1].an-1;
                St[top].flag=1;
            }
        }
        else if (St[top].flag==0)  //已计算出vf值
        {
            if (top>0&&St[top-1].an==0) //(2)式
            {
                St[top-1].af=St[top].af;
                St[top-1].flag=0;
                top--;
            }
            else if (top > 0) //(3)式
            {
                St[top-1].am=St[top-1].am-1;
                St[top-1].an=St[top].af;
                St[top-1].flag=1;
                top--;
            }
        }
        if(top==0 && St[top].flag==0) //栈中只有一个已求出vf的元素时退出循环
            break;
    }
    return St[top].af;

}
```

(10)

①求链表中最大整数

```
int GetMax(LinkList &L){
    int m;
    if(!L->next){//判断到顶，出栈
        return L->data;
    }
    m=Max(L->next);
    return m>=L->data?m:L->data;//递归调用，得出链表中的最大整数
}
```

②求链表结点个数

```
    if(!L->next) return 1;//判断到顶，出栈
    else return (GetLength(L->next));//递归调用，得出链表结点的个数
}
```

①求所有整数的平均值

```
    double a;
    if(!L->next) return L->data;//判断到顶，出栈
    else{
        a=GetAverage(L->next,n-1);//前n-1个数的平均值
        return ((a*(n-1)+L->data))/n;//递归调用，得出n个数的平均值
    }
```