3.1 设A、B、C、D、E五个元素以此进栈（进栈后可立即出栈），问能否得到下列序列，若能得到，则给出相应的push和pop序列；若不能，则说明理由。

（1）A, B, C, D, E

（2）A, C, E, B, D

（3）C, A, B, D, E

（4）E, D, C, B, A

答： (2)和(3)不能。对(2)中的E，B，D 而言，E 最先出栈则表明，此时B 和D 均在栈中，由于，B 先于D 进栈，所以应有D 先出栈。同理(3)也不能。

(1)能，操作序列:push(A),pop(),push(B),pop(),push(C),pop(),push(D),pop(),push(E),pop()

(4)能，操作序列:push(A),push(B),push(C),push(D),push(E),pop() ,pop(),pop(),pop(),pop()

3.3 利用栈计算下列表达式的值，画出栈中元素的变化过程，并指出栈中最多时有几个元素。

(1) 5 3 2 * 3 + 3 / +

8 最多3个元素

（2）4 2 4 1 * 1 3 * - ^ 2 1 * / +

5 最多 5 个元素 3.15

3.4 写出下列表达式的后缀形式

(1) (a+b)/(c+d)	*ab+cd+/*
(2) b^2-4*a*c	*b2^4a*c*-*
(3) a*c-b/c^2	*ac*-bc2^/-*
(4) (a+b)*c+d/(e+f)	*ab+c*def+/+*
(5) (a+b)*(c*d+e)-a*c	*ab+cd*e+*ac*-*

3.5 利用栈将下列表达式转换为后缀形式，画出栈中元素的变化过程及各次结果

（2）a+b*(c-d)-e/f

abcd-*+ef/-

3.13　编程实现利用队列将栈中元素逆置的算法。

算法思想：
（1）创建一个队列；
（2）将栈中元素移入队列；
（3）将队列中元素移入堆栈。

```
template <class T>
void SeqStack<T>::Invert(void)
{
    SeqQueue<T> sq(maxTop + 1);
    T temp;
    while (top != -1)
    {
          this->Top(temp);
          this->Pop();
          sq.EnQueue(temp);
     }
     while (!sq.IsEmpty())
     {
          sq.Front(temp);
          sq.DeQueue();
          this->Push(temp);
      }
}
```

3.15 (1) 对整数数组 A[n]设计递归算法求数组中的最大整数。
算法思想：n 元数组最大值等于数组最末元素与前 n-1 个元素数组最大值中的较大者。

假设当前数组 A 含有 n 个元素，则：
（1） 如果 n=1，即：当前数组只包含一个元素，则直接返回该元素为最大值。
（2） 如果 n>1，即：当前数组中有多于一个元素，则：
（1.1）递归计算当前数组前 n-1 个元素中的最大整数 maxInt；
（1.2）返回当前数组最末位元素 A[n-1]与 maxInt 中的最大值；

```
int max (int A[ ], int n) //n 表示数组元素个数
{
    if (n==1)
        return A[0];
    else
        maxInt = maxIntArray(A, n-1);
        if (A[n-1] > maxInt
            return A[n-1];
        else
            return maxInt;
}
```

或：

```
int maxIntArray (int A[ ], int n) //n 为数组最末位元素下标索引，表示数组含有 n+1 个元素
{
    if (n==0)
        return A[0];
    else
        return max (A[n], maxIntArray(A, n-1));
}
```

或：

```
int max(int a[],int i, int n)
{
    int k;
    if (i<n)
    {
        k=max(a, i+1,n);
        if (a[i]>k)
            return a[i];
        else
            return k;
    }
    else
       return a[n];
}
```

(2)求数组中 n 个数的平均值

```
int av(int a[],int n)
{
    if (n>0)
    {
       return (a[n-1]+av(a, n-1)*(n-1))/n;
    }
    return 0;
}
```