```
(1)  sum = 0;
     for( i = 0; i < n; i++ )
         sum++;

(2)  sum = 0;
     for( i = 0; i < n; i++ )
         for( j = 0; j < n; j++ )
             sum++;

(3)  sum = 0;
     for( i = 0; i < n; i++ )
         for( j = 0; j < n * n; j++ )
             sum++;

(4)  sum = 0;
     for( i = 0; i < n; i++ )
         for( j = 0; j < i; j++ )
             sum++;

(5)  sum = 0;
     for( i = 0; i < n; i++ )
         for( j = 0; j < i * i; j++ )
             for( k = 0; k < j; k++ )
                 sum++;

(6)  sum = 0;
     for( i = 1; i < n; i++ )
         for( j = 1; j < i * i; j++ )
            if( j % i == 0 )
               for( k = 0; k < j; k++ )
                  sum++;
```

2.8 假设需要生成前 N 个整数的一个随机置换。例如，{4,3,1,5,2}和{3,1,4,2,5}就是合法的置换，但{5,4,1,2,1}则不是，因为数 1 出现两次而数 3 却没有。这个程序常常用于模拟一些算法。我们假设存在一个随机数生成器 r，它有方法 randInt(i,j)，它以相同的概率生成 i 和 j 之间的整数。下面是三个算法：

1. 如下填入从 a[0]到 a[n-1]的数组 a；为了填入 a[i]，生成随机数直到它不同于已经生成的 a[0],a[1],…,a[i-1]时再将其填入 a[i]。
2. 同算法(1)，但是要保存一个附加的数组，称之为 used 数组。当一个随机数 ran 最初被放入数组 a 的时候，置 used[ran]=true。这就是说，当用一个随机数填入 a[i]时，可以用一步来测试是否该随机数已经被使用，而不是像第一个算法那样(可能)用 i 步测试。
3. 填写该数组使得 a[i]=i+1。然后

```
for( i = 1; i < n; i++ )
    swapReferences( a[ i ], a[ randInt( 0, i ) ] );
```

a. 证明这三个算法都生成合法的置换，并且所有的置换都是可能的。

b. 对每一个算法给出你能够得到的尽可能准确的期望运行时间分析(用大 O)。

c. 分别写出程序来执行每个算法 10 次，得出一个好的平均值。对 $N=250,500,1000,2000$ 运行程序(1)；对 $N=25\,000,50\,000,100\,000,200\,000,400\,000,800\,000$ 运行程序(2)；对 $N=100\,000,200\,000,400\,000,800\,000,1\,600\,000,3\,200\,000,6\,400\,000$