第 1 题 正误判断（凡交代未尽之处，皆以讲义及示例代码为准）

1. （ ）对有序向量做 Fibonacci 查找，就最坏情况而言，成功查找所需的比较次数与失败查找相等。

2. （ ） $f(n)=O(g(n))$，当且仅当 $g(n)=\Omega(f(n))$。

3. （ ）若借助二分法查找确定每个元素的插入位置，向量的插入排序只需时间 $O(n\log n)$ 时间。

4. （ ）RPN 中各操作数的相对次序，与原中缀表达式完全一致。

5. （ ）对不含括号的中缀表达式求值时，操作法栈的容量可以固定为某一常数。

6. （ ）无论有序向量或有序列表，最坏情况下均可在 $O(\log n)$ 时间内完成一次查找。

7.（ ）只要是采用基于比较的排序算法，对任何输入序列都至少需要运行 $\Omega(n\log n)$ 时间。

8. （ ）对于同一有序向量，每次折半查找绝不会慢于顺序查找。

第 2 题 多重选择

1. （ ）共有几种栈混洗方案，可以使字符序列{‘x’,’o’,’o’,’o’,’x’}的输出保持原样？

   A．12　　B. 10　　C. 6　　D. 5

2. （ ）若 $f(n)=O(n^2)$ 且 $g(n)=O(n)$，则下列结论正确的是：

   A. $f(n)+g(n)=O(n^2)$　　B. $f(n)/g(n)=O(n^2)$　　C. $g(n)=O(f(n))$　　D. $f(n)*g(n)=O(n^3)$

3. （ ）对长度为 $n=Fib(k)-1$ 的有向序列做 Fibonacci 查找。若个元素的数值等概率独立均匀分布，且平均成功查找长度为 L，则失败平均查找长度为：

   A．n(L-1)/(n-1)　　B. n(L+1)/(n+1)　　C. (n-1)L/n　　D. (n+1)L/n

4. （ ）对长度为 Fib(12) – 1 = 143 的有序向量做 Fibonacci 查找，比较操作的次数至多为：

   A．12　　B. 11　　C. 10　　D. 9

5. （ ）算法 g(n)的复杂度为 $\Theta(n)$。若算法 f(n)中有 5 条调用 g(n)的指令，则 f(n)的复杂度为：

   A．$\Theta(n)$　　B．$O(n)$　　C．$\Omega(n)$　　D．不确定

第 3 题 估计以下函数 F(n)的复杂度（假定 int 类型字长无限，且递归不会溢出）

void F(int n)　　//O(　　)
{
　for (int i = 0, j = 0; i<n; i+=j, j++);
}
	void F(int n)　　//O(　　)
	{
	　for (int i = 1, r = 1; i<n; i<<=r, r<<=1);
	}

void F(int n)　　//O(　　)
{
　for (int i=1; i<n; i++)
　　for (int j=0; j<n; j+=i);
}
	void F(int n) //expected-O(　　)
	{
	　for (int i=1; i<n; i++)
	　　if(0 == rand()%i)
	　　　for (int j=1; j<n; j<<=1);
	}

```
void F(int n)                //O(          )
{
      for (int i=1; i<n; i=1<<i);
}
```

```
void F(int n)                //O(          )
{
      return (n<4) ? n : F(n>>1)+F(n>>2);
}
```

```
int F(int n)                 //O(               )
{
      return (n==0) ? 1 : G(2, F(n-1));
}
int G(int n, int m)
{
      return (m==0) ? 0 : n+G(n, m-1);
}
```

```
int F(int n)                 //O(             )
{
      return G(G(n-1));
}
int G(int n)
{
      return (n==0) ? 0 : G(n-1)+2*n-1;
}
```

第 4 题　分析与计算

1. 考察如下问题：任给 12 个互异的整数，且其中 10 个已组织为一个有序序列，现需要插入剩余的两个已完成整体排序。若采用基于比较的算法（CBA），最坏情况下至少需要做几次比较？为什么？

2. 向量的插入排序由 n 次迭代完成，逐次插入各元素。为插入第 k 个元素，最坏情况需要做 k 次移动，最好情况则无需移动。从期望的角度来看，无需移动操作的迭代次数平均有多少次？为什么？
   假定个元素是等概率独立均匀分布的。

3. **现有一长度为 15 的有序向量 A[0…14]，个元素被成功查找的概率如下**：

I	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
$P_i(\sum =$	1/128	1/128	1/32	1/8	1/8	1/32	1/16	1/16	1/128	1/64	1/16	1/4	3/16	1/128	1/64

   若采用二分查找算法，试计算该结构的平均成功查找长度。

4. 考察表达式求值算法。算法执行过程中的某时刻，若操作符栈中的括号多达 2010 个，则此时栈的规模（含栈底的'\n'）至多可能多达？试说明理由，并示范性地画出当时栈中的内容。

5. 阅读以下程序，试给出其中 ListReport()一句的输出结果(即当时序列 L 中个元素的数值）

```
#define LLiST_ELEM_TYPE_iNT        //节点数据域为 int 型
LvalueType visit(LvalueType e)
{
        static int lemda = 1980;
        lemda += e*e;
```

```
        return lemda;
    }
    int main(int argc, char* argv[])
    {
        LList* L = Listinit(-1);
        for(int i=0; i<5; i++)
            ListinsertLast(L, i);
        ListTraverse(L, visit);
        ListReport(L);
*/
        ListDestroy(L);
        return 0;
    }
```

第 5 题　基于 **ADT** 操作实现算法（如有必要，可增加子函数）

1、sortOddEvev(L)

```
#define LLiST_TYPE_ARRAY       //基于向量实现序列
#define LLiST_ELEM_TYPE_iNT    //节点数据域为 int 型
/*****************************************************************
*输入：基于向量实现的序列 L
*功能：移动 L 中元素，使得所有奇数集中于前端，所有偶数都集中于后端
*输出：无
*实例：L = {2，13，7，4，6，3，7，12，9}，则排列序后
*      L = {9，13，7，7，3，6，4，12，2}
*要求：O(n)时间，O(1)附加空间
*****************************************************************/
void sortOddEvev(LList* L){
```

}

2、shift(L,K)

#define LLiST_TYPE_ARRAY

#definr LLiST_ELEM_TYPE_iNT

/***

*输入：基于向量实现的序列 L

*功能: 将 L 中各元素循环左移 k 位

*输出：无

*实例：L = {1, ..., k, k+1, ..., n}，则左移后

*   L = {k+1, ..., n, 1, ..., k}

*要求：O(n)时间（注意：最坏情况下 k=Ω（n）），O(1)附加空间

***/

void shift(LList* L, int k) {   // Assert: L != NULL. 0 < k < Length(L) 【Assert：保证(数据满足)】

}