## 一、 选择题（每题 2 分，共 30 分）

1. 程序在计算机上的运行时间（ ）。

A. 与所处理的操作对象的数据结构无关

B. 与所使用的程序设计语言无关

C. 与问题的规模相关

D. 与程序采用的算法的策略无关

2. 设 n 是表示问题规模的非负整数，以下算法的时间复杂度和空间复杂度分别为（ ）。

```
int fun(int n){
    int i=1,s=1;
    while (i<n){
        s++;
        i = 3*i;
    }
    return s;
}
```

A. $O(\log_3 n)$, O(n)　　B. $O(\log_2 n)$, O(1)

C. $O(\log_2 n)$, O(n)　　D. $O(n\log_3 n)$ , O(1)

3．栈和队列具有相同的（ ）。

A. 抽象数据类型　　B. 逻辑结构

C. 存储结构　　D. 基本操作

4. 设栈 S 的初始状态均为空，整数 0 到 9 依次进入栈 S。入栈和出栈可以交替进行，当一个元素被 pop 时，它被输出到终端。下列序列中不可能为输出序列的是（ ）。

A. 0 1 2 3 4 5 6 7 8 9

B. 4 3 2 1 0 9 8 7 6 5

C. 4 3 2 1 0 5 6 7 8 9

D. 5 6 7 8 9 0 1 2 3 4

5. 利用栈 S 对以下后缀表达式求值: 8 2 3 ^ / 2 3 * + ，其中 ^ 是求幂运算符。在 * 被计算后，S 中栈顶的两个元素是（ ）。

A. 6、1　　B．5、7　　C. 3、2　　D. 1、5

6. 假设用一个长度为 n 的数组表示一个容量为 n - 1 的循环队列，入队和出队分别使用 rear 和 front 作为数组索引变量进行操作，初始为空时 rear = front = 0，则判别队列满和队列空的条件为（ ）。

A. 满：(rear+1) % n == front，空：rear == front

B. 满：(front+1) % n == front，空：(front+1) % n == rear

C. 满：rear == front，空：(rear+1) % n == front

D. 满：(front+1) % n == rear，空：rear == front

7.以下属于线性结构的是（ ）。

A .由 *n* 个实数组成的集合

B. 由所有实数组成的集合

C. 由所有整数组成的序列

D. 由 *n* 个字符组成的序列

8. 顺序存储结构与链式存储结构比较时，正确的是（　　）。

A. 顺序实现时数组的大小必须预先确定，而链式实现时链表可以随时改变大小。

B.在链表中插入和删除操作更容易

C.在典型的链式实现中不允许随机访问

D.以上都对

9. 给定指向单链表头结点和尾结点的两个指针，时间性能与表长相关的操作是（　　）。

A. 删除首元素

B .在首位置插入一个新元素

C. 在链表尾部插入一个新元素

D. 删除表的最后一个元素

10. 在（　）存储结构下，两个线性表的连接操作时间复杂度为 O(1)。如 A=（1,3,2），B=（5,4），A 和 B 的连接为（1,3,2,5,4）。假设各个链表都仅设头指针。

A. 单链表

B. 双链表

C. 双向循环链表

D. 顺序表

11. C++语言中向量 vector 的本质是（　　）。

A. 顺序栈　　B. 单链表

C. 双端队列　　D. 动态顺序表

12. 以下算法的时间复杂度为（　）。

```
int f(int n) {
   if (n < 3)
      return 0;
   int num = 0;
   for (int i = 1; i <= n; i++)
      num += 1;
   int k = n;
   while (k >= 1) {
      k = k / 2;
      num += 1;
      }
   return f(n-2) + num ;
}
```

A. $O(n)$　　B. $O(n^2)$　　C. $O(n\log_2 n)$　　D. $O(\log^2 n)$

13. 第 12 题中算法的空间复杂度为（ ）。

A. O(n)　　B. $O(n^2)$　　C. $O(n\log_2 n)$　　D. $O(\log^2 n)$

14. KMP 算法中用 next 数组存放模式串的部分匹配信息，当模式串 j 位置的字符与主串 i 位置的字符不等时，i 的变化方式是（ ）。

A. i 不变　　B. i = j + 1

C. i = i – j +1　　D. i = j – i +1

15. 设模式 T="abcabc"，则在 KMP 算法中该模式的 next 值为（ ）。

A. {-1,0,0,1,2,3}　　B. {-1,0,0,0,1,2}

C. {-1,0,0,1,1,2}　　D. {-1,0,0,0,2,3}

## 二、 判断题（打√或打×，每题 2 分，共 20 分）

1. Sequentially stored linear lists do not support random access.

2. 不论是入队操作还是入栈操作，在顺序存储结构上都需要考虑“溢出”情况。

3. 对于同一种逻辑结构，同一个运算在不同的存储方式下实现，其运算效率可能不同。

4. 在顺序队列的物理模型表示法中，入队操作和出队操作的时间复杂度均是 O(1)。

5. 在顺序表中插入一个元素，移动元素的次数仅与插入位置有关。

6. 可以通过链队列中的队首、队尾指针计算出队列中元素的个数。

7. 对于一个特定的算法，同样的输入可能产生不同的输出。

8. 后缀、中缀和前缀表达式中，操作数的出现次序总是一致的。

9. 对一个不含相同元素的递增有序的数组，找出第 i 大的元素的操作的时间复杂度是 O(1)。

10. 线性表采用链式存储表示时，所有结点之间的存储单元地址可连续可不连续。

## 三、 应用题（共 26 分）

1. （1）假设一个算法在输入规模 n 为 1000、2000、3000 和 4000 时的运行时间分别为 5 秒、20 秒、45 秒和 80 秒。请估计当 n 为 5000 时算法的运行时间，并用大 O 记号表示该算法的时间复杂度。

（2）随着问题规模的不断扩大，同一算法所需的计算时间通常都呈单调递增趋势，但情况也并非总是如此。试举例说明，随着问题规模的扩大，同一算法所需的计算时间可能上下波动。（6 分）

2. （1）简要说明线性表、栈、顺序栈这三个概念之间的区别和联系。

（2）举例说明，在含有圆括号和花括号的文本中如何检查所有括号是否配对。

（3）解释如何使用两个栈实现一个队列。（10 分）

3. （1）简述用单链表实现线性表的优缺点和使用场合。

（2）简述在单链表中设置头结点的作用。

（3）若需对一元多项式进行存储并主要进行加法和乘法运算，分析多项式的逻辑结构和存储结构。（10 分）

## 四、 算法题（共 24 分）

1. 假设用带头结点的单链表 LinkList 类作为线性表的存储结构，LinkList 类的数据成员只含有表头结点指针。为 LinkList 类添加一个方法，将链表向左循环移动 k 次。当 k 的值不合法，则返回 false，否则返回 true。要求算法的时间复杂度和空间复杂度分别为 O(n)和 O(1)。

例如：原表为(1,2,3,4,5)，当 k 为 2 或 7 等时，则向左循环移动 k 次后，表变为(3,4,5,1,2)，返回 true。如 k<0，则返回 false。（8 分）

请参考以下原型进行算法设计：

```
template <class DataType>
bool LinkList<DataType>::shift(int k);
```

2. 假设用顺序表 SeqList 类实现递增有序表，为 SeqList 类添加一个方法，删除递增有序表中的重复元素，如原表为（1,2,2,3,4,4,6），则删除重复后的新表为（1,2,3,4,6）。要求算法的时间复杂度和空间复杂度分别为 O(n)和 O(1)。（8 分）

请参考以下原型进行算法设计：

```
template <class DataType>
void SeqList<DataType>::delete_dup( );
```

3. 设计递归算法，通过 head 为首结点指针的整数单链表，生成一个包含 head 链表中的所有奇数值的新链表。如原链表为 1->2->3->5->10，则新表为 1->3->5。要求原链表保持不变，算法返回新链表的首结点指针。（8 分）

请参考以下原型进行算法设计：

```
Node<int> *LinkList<int>:: delete_odd(Node<int> *head)
```