1. 数组 Q[n]用来表示一个循环队列，front 为队头元素的前一个位置，rear 为队尾元素的位置，计算队列中元素个数的公式为_____。

答案： (rear-front+n)%n
解答：循环队列中，rear-front 的结果可能是负整数，而对一个负整数求模，其结果在不同的编译器环境中可能会有所不同。

2. 二维数组 A 中行下标是 10～20，列下标是 5～10，按行优先存储，每个元素占 4 个存储单元，A[10][5]的存储地址是 1000，则元素 A[15][10]的存储地址是_____。

答案： 1140
解答：数组 A 中每行共有 6 个元素，元素 A[15][10]的前面共存储了（15-10）×6+5 个元素，每个元素占 4 个存储单元，所以，其存储地址是 1000+140=1140。

3. 一个 n×n 的对称矩阵，按列优先进行压缩存储，则其存储容量为_____。

答案： n (n+1)/2
解答： 对于对称矩阵，行优先和列优先，其存储容量是一样的，1+2+…+n = n (n+1)/2。

4. 带头结点的循环链表 L 为空表的条件是_____。

答案： L->next=L
解答：循环链表没有空指针，循环链表为空时只有一个头结点，因此，头结点的前驱指针指回头结点：L->next=L。

5. 对于由 n 个元素组成的线性表，建立一个单链表的时间复杂度是_____。

答案： O(n)

6. 在一个单链表中，已知 q 所指结点是 p 所指结点的直接前驱，若在 q 和 p 之间插入 s 所指结点，则需要执行下列操作：_____。

答案： q->next=s; s->next=p;

7. 已知一个栈的进栈序列是 1，2，3，…，n，其输出序列是 p1，p2，…，pn，若 p1=n，则 pi 的值是_____ 。

答案： n-i+1
解答：当 p1=n 时，输出序列是唯一的，即为 n、n-1、…、2，1，则 p2=n-1，…，pn=1，推断出 pi=n-i+1。

8. 设目标串为 s=“abcababababaab”，模式串为 p=“babab”，则 KMP 模式匹配算法的 next 数组为_____ 。

答案: {-1,0,0,1,2}

解答: next 数组只与模式串 p 有关，计算过程如下表所示:

j	0	1	2	3	4
p	b	a	b	a	b
next[j]	-1	0	0	1	2

9. 下列求解汉诺塔问题的递归程序，其时间复杂度为_____ 。

```
void hanoi(int n, char a, char b, char c)
{
    if(n==1)
        cout<<"a-c";
    else
    {
        hanoi(n-1,a,c,b);
        cout<<"a-c";
        hanoi(n-1,b,a,c);
    }
}
```

答案: $O(2^n)$

解答: 从上述递归关系中可以看出

$$T(n)=\begin{cases}1 & n=1\\ 2T(n-1)+1 & n>1\end{cases}$$

所以有

$T(n) = 2T(n-1)+1$

$=2 (2T(n-2)+1) +1 = \mathbf{2^2\ T(n-2)}+2^1+1$

$=2^2 (2T(n-3)+1)+2^1+1 = \mathbf{2^3\ T(n-3)}+2^2+2^1+1$

$=\cdots\cdots$

$=2^{n-1} T(1)+2^{n-1}+2^{n-2}+\cdots\cdots+2^0$

$=O(2^{n-1})$

$= O(2^n)$

10. 已知单链表中各结点的元素值为整数且递增有序，设计算法"DeleteBetween"删除链表中所有大于 mink 且小于 maxk 的元素，并释放被删结点的存储空间。

答案:

```
DeleteBetween(Node<T> *first, int mink, int maxk)
{
    p=first;
    while(p->next!=NULL&&p->next->data<=mink)
      p=p->next;
    if(p->next!=NULL)
      q=p->next;
```

```
        while(q->data<maxk)
        {
          u=q->next;
          p->next=q->next;
          delete q;
          q=u;
         }
}
```

解答： 因为是在有序单链表上的操作，所以，要充分利用其有序性。在单链表中查找第一个大于 mink 的结点和第一个小于 maxk 的结点，再将二者间的所有结点删除。

11. 给定一个链表，判断链表是否存在环型链表。

答案:

```
struct link {
    int data;
    link* next;
};

bool IsLoop(link* head)
{
    link* slow=head, *fast = head;
    if (head ==NULL || head->next ==NULL) {
        return false;
    }
    do{
        slow= slow->next;
        fast= fast->next->next;
    } while(fast && fast->next && slow!=fast);
    if (slow ==fast)
        return true;
    else
        return false;
}
```

解答： 判断链表是否存在环型链表问题：判断一个链表是否存在环，例如下面这个链表就存在一个环：N1->N2->N3->N4->N5->N2 就是一个有环的链表，环的开始结点是 N5 这里有一个比较简单的解法。设置两个指针 slow，fast。每次循环 slow 向前走一步，fast 向前走两步。直到 fast 碰到 NULL 指针或者两个指针相等结束循环。如果两个指针相等则说明存在环。