复习 1

一、

1.下面程序的时间复杂性的量级为（ ）。

```
int i=0，s1=0，s2=0；
while（i++<n）
  { if (i%2) s1+=i;
    else s2+=i;
  }
```

A. O(1)　　B. O(lbn)　　C. O（n）　　D. O(2n)

2. 设目标串 T=“abccdcdccbaa”，模式串 P=“cdcc”，则朴素的模式匹配追踪算法在第 （ ）趟匹配成功。

A. 5　　B. 6　　C. 4　　D. 3

3. 在一个顺序表中的任何位置插入一个元素的时间复杂度为（ ）。

A. O(n)　　B. O(lbn)　　C. O(1)　　D. $O(n^2)$

4. 线性表的链式存储比顺序存储更有利于进行（ ）操作。

A.查找　　B.表尾插入和删除

C.按值插入和删除　　D.表头的插入和删除

5. 在一个带头结点的循环双向链表中，若要在 P 所指向的结点之前插入一个新结点，则需要相继修改（ ）个指针域的值。

A.2　　B.3　　C.4　　D.6

6. 一个表头指针为 ph 的带头结点的单链表，若要向表头插入一个由指针 p 指向的结点，则应执行（ ）操作。

A. ph=p; p->next=ph;　　B. p->next=ph; ph=p;

C. p->next=ph; p=ph;　　D. p->next=ph->next; ph->next=p;

7. 设有一个 15 阶的对称矩阵 A，采用压缩存储的方式，将其下三角部分以行序为主序存储到一维数组 B 中（数组下标从 1 开始），则矩阵中元素 a7,6 在一维数组 B 中的下标是（ ）。

A. 42　　B. 13　　C. 27　　D. 32

8. 在一棵深度为 h 的完全二叉树中，所含结点个数不大于（ ）。

A.2^h　　B.2^{h+1}　　C.2^h-1　　D.2^{h-1}

9. 设有一顺序栈 S，元素 s1,s2,s3,s4,s5,s6 依次进栈，如果 6 个元素出栈的顺序是 s2,s3,s4,s6 ， s5,s1,则栈的容量至少应该是 （ ）。

A. 2　B. 3　C. 5　D. 6

10. 若数据元素序列 11，12，13，7，8，9，23，4，5 是采用下列排序方法之一得到的 第二趟排序后的结果，则该排序算法只能是（　　）。

A．起泡排序　B．插入排序 C．选择排序 D．二路归并排序

11. 在一个具有 n 个顶点的有向图中，若所有顶点的出度数之和为 s，则所有的入度数之和为（　　）。

A. s　　B. s-1　　C. s+1　　D. n

12. 在一个具有 n 个顶点的无向图中，若具有 e 条边，则所有顶点的度数为（　　）。

A. n　　B. e　　C. n+e　　D. 2e

13. 若有一个图中包含 k 个连通分量，若按照深度优先搜索的方法访问所有顶点，则必须调用（　　）次深度优先搜索遍历的算法。

A．k　　B．1　　C．k-1　　D．k+1

14. 已知一棵完全二叉树第 8 层只有 9 个结点，则该二叉树共有（　　）个叶子结点。

A．65　B．66　C．67　D．68

15. 对于一个有向图，若一个顶点的度为 k1，出度为 k2，则对应逆邻接表中该顶点单链表的边数结点为（　　）。

A．k1　B．k2　C．k1-k2　D．k1+k2

16. 若一个图的边集为{（A，B）（A，C）（B，D）（C，F）（D，E）（D，F）}，则从顶点 A 开始对该图进行深度优先搜索，得到的顶点序列可能为（　　）。

A．ABCFDE　　B．ACFDEB　C．ABDCFE　　D．ABDFEC

17. 在有序表{1，3，8，13，33，42，46，63，76，78，86，97，100}中，用折半查找值 86 时，经（　　）次比较后查找成功。

A．6　　B．3　　C．8　　D．4

18. 在排序过程中，可以通过某一趟排序的相关操作所提供的信息，判断序列是否已经排好序，从而可以提前结束排序过程的排序算法是(　　)。

A．冒泡　　B．选择　　C．直接插入　　D．折半插入

19. 排序方法中，从尚未排序序列中挑选元素，并将其依次放入已排序序列（初始为空）的一端的方法，称为（　　）排序。

A．归并　　B．插入　　C．选择　　D．快速

20. 若一棵具有 n（n>0）个结点的二叉树的先序序列与后序序列正好相反，则该二叉树一定是（　　）。

A. 高度为 n 的二叉树　　B. 结点均无右孩子的二叉树

C. 结点均无左孩子的二叉树　　D．存在度为 2 的结点的二叉树

二、

1. 若用一个大小为 6 的数组来实现循环队列，且当 rear 和 front 的值分别为 0 和 3。当从队列中删除一个元素，再加入两个元素后，rear 和 front 的值分别是_____。

2. 在以 HL 为头指针的带头结点的单链表和循环单链表中，链表为空的条件分别为_____和_____。

3. 在一个单链表中指针 p 所指向结点的后面插入一个指针 q 所指向的节点时，首先把_____的值赋给 q->next，然后把_____的值赋给 p->next。

4. 双向链表中指针 p 所指向的结点之后插入一个指针 q 所指向的结点时，需对 p->next->prior 指针域赋值_____。

5. 设元素 a，b，c，d 依次进栈，若要在输出端得到序列 cbda，则应进行的操作序列为 Push(s,a),Push(s,b),Push(s,c),_____,_____,_____,Pop(s),Pop(s)。

6. 一个二维数组 A[15，10]采用行优先方式存储，每个数据元素占用 4 个存储单元，以该数组第 3 列第 0 行的地址（即 A[3，0]的地址）1000 为首地址，则 A[12，9]的地址为_____。

7. 有一个 8×8 的下三角矩阵 A，若将其进行按行优先存储，每个元素占用 4 个字节，则 $a_{5,4}$ 元素的相对字节地址为（相对首元素地址而言）_____。

8. 假定一棵二叉树顺序存储在一维数组 a 中，若 a[5]元素的左孩子存在则对应的元素为_____，若右孩子存在则对应的元素为_____，双亲元素为_____。

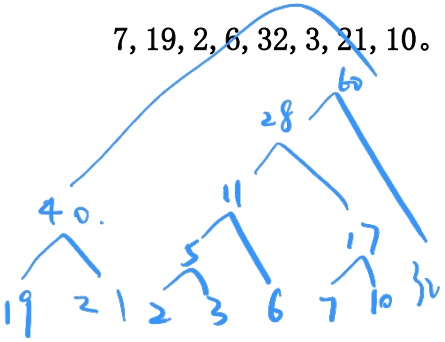
9. 若一个图的顶点集为{a,b,c,d,e,f}，边集为{(a,b)，(a,c)，(b,c)，(d,e)}，则该图含有_____个连通分量。

10. 从一棵二叉排序树中查找某个元素时，若元素的值等于根结点的值，则表明_____，若元素的值小于根结点的值，则继续向_____查找，若元素的值大于根结点的值，则继续向_____查找。

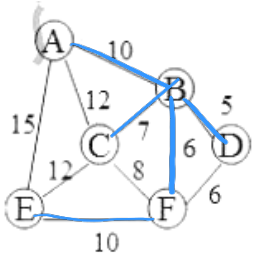
11. 假定一组记录为（46，79，56，38，40，80），对其进行快速排序的第一次划分后的结果为_____。

三、

1. 请为字符 a,b,c,d,e,f,g,h 进行哈夫曼编码，它们对应的出现次数分别为 7,19,2,6,32,3,21,10。

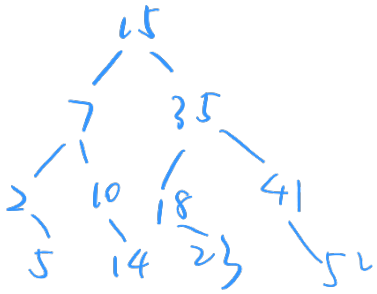


2. 一个无向图如下图所示，请写出从顶点 A 出发采用 Prim 算法最小生成树的过程。



3. 对顺序表{2、5、7、10、14、15、18、23、35、41、52} 进行折半查找，按 $mid=\left\lfloor \frac{low+high}{2} \right\rfloor$ 计算中点的位置。（1）画出折半查找判定树；（2）计算查找 15、7、14、12 所需的比较次数分别是多少？

1 2 4 4



4. 假设以带头结点的单链表表示线性表，阅读以下算法，回答问题：

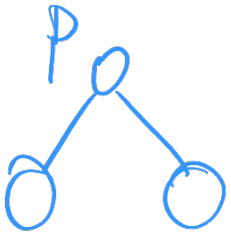
（1）线性表为（a1, a2, a3, a4, a5, a6, a7），写出算法执行后的线性表；

（2）简述算法的功能；

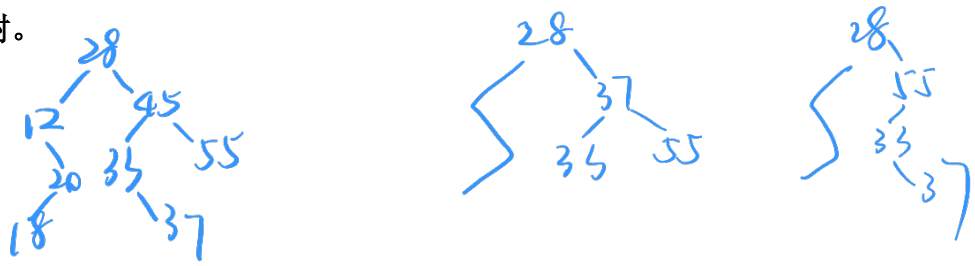
```
void function(linklist *L)
 { // L 为带头结点的单链表头指针
    linklist *p,*q;
    p=L;
    while (p && p->next)
     { q=p->next;
       p->next=q->next;
       p=q->next;
       free(q);
      }
    }
```

5. 以下算法是关于二叉树的运算，阅读算法，写出其功能，并在 4 个注释处写出注释内容。

```
typedef struct node
   { int data;
     struct node *lchild, *rchild;
   } bitree;

 bitree *fun (bitree *root, int k)
 { bitree *p=root;
   while(p)
   if (p->data==k) break;// ①
   else if (k<p->data) p=p->lchild;// ②
        else p=p->rchild;// ③
   return p; // ④
   }
```



6. 按照关键字序列{28,45,33,12,37,20,18,55}的先后次序生成一棵二叉排序树，然后删除关键字 45，要求删除后仍然是一棵二叉排序树，如果有多种删除方法，请分别画出删除 45 后的二叉排序树。



7. 算法是关于数据表 r 的操作，如果数据表存放的 5 个数据为 5，4，11，2，9，试分析算法的功能及算法执行后 r 的最终结果。（备注：r[0]单元闲置，从 r[1]开始存放数据表中元素。）

```
typedef int datatype
void function(datatype r[], int n)
  { int i, j, low, high, mid;
    for(i=2; i<=n; i++)
    { r[0]=r[i] ;
       low=1;
       high=i-1;
       while(low<=high)
       { mid=(low+high)/2;
         if (r[0]>r[mid]) high=mid-1;
         else low=mid+1;
        }
       for(j=i-1; j>=low; j--)
         r[j+1]=r[j];
      r[low]=r[0];
       }
  }
```

8. 阅读算法，并回答下列问题：

（1）设队列 Q=（2，4，6，8，10，12），写出执行算法 test 后的队列 Q 中的元素；

（2）简述算法 test 的功能。

```
void  test(Queue *Q){
    datatype  e;
    if (!QueueEmpty(Q)){
      e=DeQueue(Q);
      test(Q);
      EnQueue(Q,e);
    }
}
```

9. 已知图 g 具有 n 个顶点 e 条边，按照以下关于图的操作算法（算法参数 k 的值为 3），以及顶点表和邻接矩阵的内容，回答问题：

（1） 写出算法的输出结果；

（2） 根据输出的顶点和边，画图表示算法的结果；

（3） 说明算法的功能。

顶点表和邻接矩阵如下：

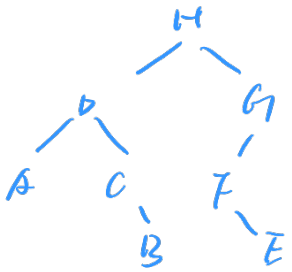
$$g->vexs=\begin{bmatrix}A\\B\\C\\D\end{bmatrix}\qquad g->arcs=\begin{bmatrix}0&1&1&0\\1&0&1&1\\1&1&0&0\\0&1&0&0\end{bmatrix}$$

图的操作算法如下：

```
typedef  struct
{  char   vexs[n+1];  //顶点数组
   int   arcs[n+1][n+1];  //邻接矩阵
}graph;
```

```
graph*g=(graph*)malloc(sizeof(graph));
void g_op(int k)
{  SETNULL(Q);       //置 Q 为空队
    for(i=1; i<=n; i++)   //辅助数组清零
      visited[i]=0;
    printf(“%c”, g->vexs[k]);
    visited[k]=1;
    ENQUEUE(Q, k);  // k 入队
    while( !EMPTY(Q) ) //队列非空执行循环体
      {  i=DEQUEUE(Q);     //出队
        for( j=1; j<=n; j++)
         if((g->arcs[i][j]==1)&&( visited[j]!=1))
          {  printf(“(%d,%d)”,i,j);
             printf(“%c”, g->vexs[j]);
             visited[j]=1;
             ENQUEUE(Q, j);   //j 入队
           }
        }
}
```

10. 已知二叉树的先序序列和中序序列分别为 HDACBGFE 和 ADCBHFEG，画出该二叉树。



11.

1. 假设有一带头结点的非空单链表 HL，试着设计一函数 Count_Del( )将单链表 HL 中所有为 x 的结点删除，并统计出结点值为 x 的结点数占总结点数的百分比，并将统计结果返给主调函数。要求算法的时间复杂度为 O(n) ,释放被删除结点所占空间。

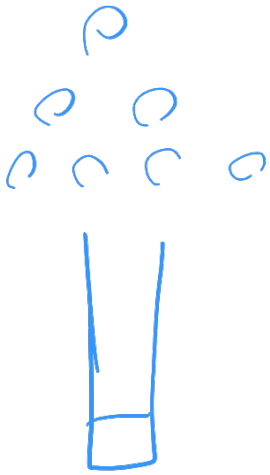
```
typedef struct node
  {  char data;
     struct node *next;
  }linklist; //单链表结点类型定义

double Count_Del (linklist* HL, char x)
{
```

double cnt = 0, num = 0;
linklist* prev = HL, current = HL->next;
while (!current->next) num++;
if (current->data == 'x')
{ cnt++

```
}
```

12. 以下算法是关于二叉树的先序遍历非递归算法实现，请在①~④处将下面的算法补充完整。

```
typedef struct node
```



```
{  int key;
   struct node *lchild,*rchild;
}bitree;//二叉树结构类型定义
void npreorder(bitree *t)
{  bitree *p = t;
   bitree* Q[maxsize];//用数组定义顺序栈
   int top = -1;//顺序栈栈顶指针
   while (p != NULL || top != -1)
   {   if (p != NULL)
       { ①_____;
         top++;
         Q[top] = p;
         ②_____;
        }//endif
      else
         {   p = Q[top];
         ③_____;
         ④_____;
      }//endelse
    }//endwhile
}
```