一、选择题（每题 1 分，共 20 分）

1.设栈 S 和队 Q 的初始状态均为空，元素 abcdef 依次通过栈 S，一个元素出栈后即进队 Q，若 6 个元素出栈的序列是 cefdba，则栈 S 的容量至少应该是_____B

A. 6　　B. 4　　C. 3　　D. 2

2.用某种排序方法对线性表{4,9,3,7,1,5,8,6,2}进行排序时，元素序列的变化情况如下：

1）4, 9, 3, 7, 1, 5, 8, 6, 2

2）2, 1, 3, 4, 7, 5, 8, 6, 9

3）1, 2, 3, 4, 6, 5, 7, 8, 9

4）1, 2, 3, 4, 5, 6, 7, 8, 9

所采用的排序方法是（ ）C

A. 插入排序　　B. 选择排序　　C. 快速排序　　D. 二路归并排序

3.无向图 G 有 23 条边，度为 4 的结点有 5 个，度为 3 的结点有 4 个，其余都是度为 2 的结点，则图 G 有____个结点。D

A. 11　　B. 12　　C. 15　　D. 16

4.若平衡二叉树的高度为 6，且所有非叶子结点的平衡因子均为-1，则该平衡二叉树的结点总数为：B

A. 12　　B. 20　　C. 32　　D. 33

5.已知字符集{a，b，c，d，e，f，g，h}，若各字符的哈夫曼编码依次是 0100，10，0000，0101，001，011，11，0001，则编码序列 0100011001001011110101 的译码结果是：__D__

A．a c g a b f h　B．a d b a g b b　C．a f b e a g d　D．a f e e f g d

6.内排和外排的不同在于_____B_____

A. 内排数据元素的类型简单，外排数据类型复杂

B. 内排数据元素在内存中能全部放下，外排不能，一部分要存在外存储器上

C. 内排数据存储用顺序结构，外排数据存储用链式结构

D. 内排是稳定排序，外排是不稳定排序

7.以下序列不是堆的是_____ 。D

A．100,85,98,77,80,60,82,40,20,10,66

B．100,98,85,80,77,82,66,60,40,20,10

C．10,20,40,60,66,77,80,82,85,98,100

D．100,85,40,77,80,60,66,98,82,10,20

8.快速排序在下列哪种情况下最不利于发挥其长处_____。D

A. 要排序的数据量很大　　B. 要排序的数据中含有多个相同值

C. 要排序的数据个数为奇数　　D. 要排序的数据已基本有序

9.下列关于图的描述，错误的是_____。D

A．在一个无向图中，所有结点的度数之和等于所有边数的 2 倍

B．在一个有向图中，所以结点的入度或出度之和均等于所有边的数目

C. n 个结点的完全有向图包含 n(n-1)条边
D. 若要连通具有 n 个结点的无向图，至少需要 n 条边

10.用邻接表来存储图，则常见的操作的算法复杂度_____。A
A. 与图的结点数和边数都有关　　B. 只与图的边数有关
C. 只与图的结点数有关　　D. 与结点数和边数都无关

11.关于 B 树和 B+树描述中，不正确的是_____。C
A. B 树和 B+树都是平衡的多叉树
B. B 树和 B+树都可用于文件的索引结构
C. B 树和 B+树的记录均存储在叶结点上
D. B+树既支持顺序查找，又支持随机查找

12.下面的叙述中 C 是正确的。
A.若有一个结点是二叉树中某个子树的中序遍历结果序列的最后一个结点，则它一定是该子树的前序遍历结果序列的最后一个结点。
B.若有一个结点是二叉树中某个子树的前序遍历结果序列的最后一个结点，则它一定是该子树的中序遍历结果序列的最后一个结点。
C.若有一个叶子结点是二叉树中某个子树的中序遍历结果序列的最后一个结点，则它一定是该子树的前序遍历结果序列的最后一个结点。
D.若有一个叶子结点是二叉树中某个子树的前序遍历结果序列的最后一个结点，则它一定是该子树的中序遍历结果序列的最后一个结点。

13.对已十分接近按键值排序的初始序列，插入法、归并法和一般的快速分类法对其排序，算法的时间复杂度各为 C 。
A.O(N)，O(N)，O(N)　　B.O(N)，O(Nlog$_2$N)，O(Nlog$_2$N)
C.O(N)，O(Nlog$_2$N)，O(N^2)　　D.O(N^2)，O(Nlog$_2$N)，O(N^2)

14.以下程序段的时间复杂度为 A 。

```
void main() {
        int n=10, x=n, y=0;
        while (x>=(y+1)*(y+1))
             y++;
    }
```

A.O(√n)　　B.O(n)　　C.O(1)　　D.O(n^2)

15.在任意一棵非空二叉查找树 T1 中，删除某结点 v 之后形成二叉查找树 T2，再将 v 插入 T2 形成二叉查找树 T3。下列关于 T1 与 T3 的叙述中，正确的是( ) C
I. 若 v 是 T1 的叶结点，则 T1 与 T3 不同
II. 若 v 是 T1 的叶结点，则 T1 与 T3 相同
III. 若 v 不是 T1 的叶结点，则 T1 与 T3 不同
IV. 若 v 不是 T1 的叶结点，则 T1 与 T3 相同
A. 仅 I、III　　B. 仅 I、IV　　C. 仅 II、III　　D. 仅 II、IV

16.一个图有 30 个结点，这些结点的平均度数为 10，那么该图拥有多少条边？B

A.75　　B.150　　C.300　　D.600

17.图的广度优先队列遍历算法中使用队列作为其辅助数据结构，那么在算法执行过程中每个结点最多进队（）。A

A.1 次 B. 2 次 C. 3 次 D. 4 次

18.设森林中有三棵树，第一、第二和第三颗树中的结点个数分别为 m1、m2 和 m3。那么在由该森林转化成的二叉树中根结点的右子树上有（）个结点。B

A. m1+m2　　B. m2+m3　　C. m1+m3　　D. m1+m2+m3

19.在常用的描述二叉查找树的存储结构中，关键字值最大的结点（）。B

A. 左指针一定为空　　B. 右指针一定为空　　C. 左右指针均为空　　D. 左右指针均不为空

20.下面关于线性表的叙述中，错误的是 B

A、线性表采用顺序存储，必须占用一片连续的存储单元。

B、线性表采用顺序存储，便于进行插入和删除操作。

C、线性表采用链接存储，不必占用一片连续的存储单元。

D、线性表采用链接存储，便于插入和删除操作。

二、填充题（每格 1 分，共 20 分）

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**

2.对于一个有 n 个结点的图：如果是连通无向图，其边的个数至少为_____；如果是强连通有向图，其边的个数至少为____**答案：n-1，n**

3.对关键字序列{22,16,71,59,24,7,67,70,51}用 buildHeap 进行建小根堆，生成的堆按层次遍历得到的序列为_____**答案：7，16，22，51，24，71，67，70，59**

4.序列{98, 36, -9, 0, 47, 23, 1, 8, 10, 7}采用希尔排序，增量为 4 时第一趟排序结果为_____**答案：{10, 7, -9, 0, 47, 23, 1, 8, 98, 36}**

5.设待排文件 FI={16,20,4,43,9,11,55,31,28}，内存工作区容量为 w=3，按置换选择法生成的初始归并段为_____**答案：4,16,20,43,55 和 9,11,28,31**

6.将一棵树用儿子兄弟链法来表示时，树中结点的左子针指向_____，右指针指向_____。最大的儿子/第一棵子树、兄弟

7.已知一组关键字为{19,14,23,01,68,20,84,27,55,11,10,79}，按哈希函数 H(key)=key mod 13 和线性探测法处理冲突，在下表中填入散列结果

0	1	2	3	4	5	6	7	8	9	10	11	12	13

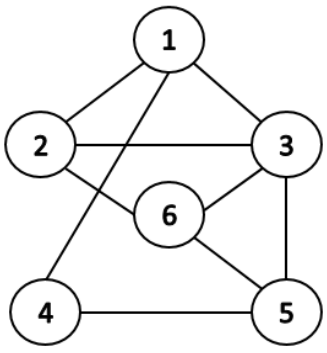
0	1	2	3	4	5	6	7	8	9	10	11	12	13
	14	01	68	27	55	19	20	84	79	23	11	10	

8.右图从结点 1 出发（假设选择原则是从小到大），深度优先遍历次序为

_____，

广度优先遍历次序为

_____。



1,2,3,5,4,6；1,2,3,4,6,5

9.在进行直接插入排序时，其数据比较次数与数据的初始排列__有__关；而在进行直接选择排序时，其数据比较次数与数据的初始排列__无__关。

10.要对图进行深度优先遍历，除了可用递归方法实现外，还可以借助__栈__的结构来实现。

11.在快速排序、堆排序、归并排序中，__归并__排序是稳定的。

12.设一棵二叉树的前序遍历序列和中序遍历序列均为 ABC，则该二叉树的后序遍历序列为 ___CBA__。

13.给出一组关键字 T=(12,2,16,30,8,28,4,10,20,6,18)，执行下列算法从小到大进行排序，并给出：
1） 冒泡排序第一趟排序结果_____。2 12 16 8 28 4 10 20 6 18 30
2）快速排序(选第一个记录为标准元素)第一趟排序结果_____。6 2 10 4 8 12 28 30 20 16 18
3）堆排序的初始化堆（最大化堆）为_____。30 20 28 12 18 16 4 10 2 6 8

14.由树的前序、后序遍历序列_____（可以/不可以）唯一确定这棵树。可以

15.中缀表达式 a+b*c+(d*e+f)*g 的后缀表达式为 abc*+de*f+g*+ 。

16.m 阶 B 树关键字个数为 m-1，继续插入一个新关键字，则分裂为两个结点，其关键字个数分别为⌈m/2⌉-1 和 m-⌈m/2⌉ 。

17.一个图中有 1000 个结点，800 条边，则其邻接矩阵有（）个矩阵元素。1000000 个

18.设图有 n 个结点和 e 条边，采用邻接表作为它的存储结构，则拓扑排序的时间复杂度为（）。O（n+e）

19.下面程序段的时间复杂度是（$O(\log_3 n)$）

```
i=0;
while(i<n) i*=3;
```

20.若某二叉树有 20 个叶子节点，有 30 个结点仅有一个孩子，则该二叉树总节点个数为（69）。

三、简答题（每题 6 分，共 12 分）

1.简述中缀表达式转换为后缀表达式的意义。写出中缀表达式 1*（2+3）/4 -5 的后缀表达式，并写出**在得到后缀表达式的过程中**读剩的表达式，运算符栈和输出的每一步变化结果

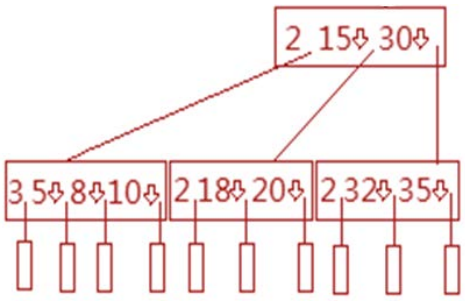
读剩的表达式	**运算符栈**	**输出**
1*（2+3）/4-5		

答案：意义提到计算机不能直接计算中缀表达式即可

后缀表达式：123+*4/5-

读剩的表达式	**运算符栈**	**输出**
1*（2+3）/4-5		
*（2+3）/4-5		1
（2+3）/4-5	*	1
2+3）/4-5	*（	1
+3）/4-5	*（	1 2
3）/4-5	*（+	1 2
）/4-5	*（+	1 2 3
/4-5	*	1 2 3 +
/4-5		1 2 3 + *
4-5	/	1 2 3 + *
-5	/	1 2 3 + * 4
-5		1 2 3 + * 4 /
5	-	1 2 3 + * 4 /
	-	1 2 3 + * 4 / 5
		1 2 3 + * 4 / 5 -

2.这是一棵 5 阶的 B 树，试问：



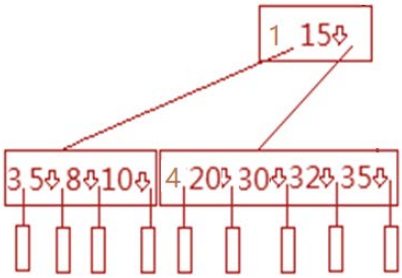
（1） 根和非根中间结点分别至少有几个孩子？

（2） 删除关键字为 18 的数据后的 B 树。

参考答案：

（1） 根至少 2 个孩子，非根中间结点至少 3 个孩子

（2）



四、分析题（12 分）

1.已知一棵二叉树是以二叉链表的形式存储的，其结点结构说明如下：

```
struct node { int data;          // 结点的数据场。
              struct node *left;    // 给出结点的左儿子的地址。
              struct node * right;   // 给出结点的右儿子的地址。
            };
```

请在 1、2 二题的 [　　] 处进行填空，完成题目要求的功能。注意，每空只能填一个语句。（8 分）

1）求出以 T 为根的二叉树或子树的结点个数。

```
int size (struct node * T ) {
        if ( [ T == NULL ] ) return 0;
        else [ return 1 +size(T->left) + size(T->right) ];
 }
```

2） 求出以 T 为根的二叉树或子树的高度。注：高度定义为树的总的层次数。

```
int height(struct node * T ) {
        if ( T == NULL ) [ return 0 ];
 else [ return 1 +( ( height(T->left) > height(T->right))?
height(T->left): height(T->right) ) ];
```

2.试分析以下算法功能及时间复杂度。（4 分）

```
typedef int ElemType; //链表数据的类型定义
typedef struct LNode{ //链表结点的结构定义
    ElemType data; //结点数据
    struct Lnode *link; //结点链接指针
    } *LinkList;
int  A(LinkList list, int k)
{
LinkList p = list->link, q = list->link;
int count = 0;
while(p != NULL)
  {  if(count < k) count++;
    else q = q->link; p = p->link;
  } //while
if(count < k) return  0;
else {
    printf(“%d”,q->data);
    return  1;
    }
}
```

答案：

查找链表中倒数第 k 个位置上的结点。若查找成功，算法输出该结点的 data 域的值，并返回 1；否则返回 0。

时间复杂度：O（n）

五、设计题（每题 8 分，共 16 分）

1.设计将数组 A[n]中的所有偶数移到奇数之前的算法。要求不增加存储空间，且时间复杂度为 O(n)。

答案：（1）设首尾两个指针 i 和 j，初始时 i=0，j=n-1。

（2）i 从前往后进行搜索，当遇到奇数时停止；j 从后往前进行搜索，当遇到偶数时停止。然后 i 和 j 指向的数据进行交换。

（3）不断进行操作（2），直到 i=j。

这个算法相当于对数组扫描一遍，复杂度为 O(n)。

2.已知二叉树，其类的定义以及二叉树结点类的定义如下所示：

```
template <class Type> class BinaryTree;   // 二叉树类前向说明
template <class Type> class BinaryNode {  // 二叉树结点类
friend class BinaryTree <Type>;
public:
    BinaryNode( ): left(NULL), right(NULL) {}  // 二叉树结点构造函数
    ~BinaryNode( ){ }
    int Degrees1(const BinaryNode <Type> * T ) const;
              // 得到以 T 为根结点的二叉树中度数为 1 的结点个数
    ……
private:
    BinaryNode <Type> *left, *right; // 结点的左、右儿子的地址
    Type data; // 结点的数据信息
};
```

设二叉树以标准形式表示，试编写一成员函数 int Degrees1(const BinaryNode <Type> * T) const; 统计二叉树中度数为 1 的结点个数。

答案：

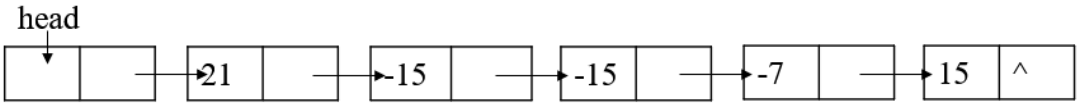
```
template <class Type>
int BinaryNode <Type>::Degrees1(const BinaryNode <Type> *T) const{
        //得到以 T 为根结点的二叉树中度数为 1 的结点个数
    if (T == NULL) return 0;
    if ( T->left != NULL && T->right == NULL || T->left == NULL && T->right != NULL)
        return 1+Degrees1(T->left)+Degrees1(T->right);
    else
        return Degrees1(T->left)+Degrees1(T->right);
}
```

六、程序题（每题 10 分，共 20 分）

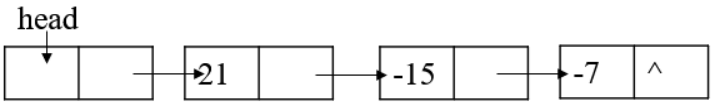
1.用单链表保存 m 个整数，结点的结构为

data	link

且|data|≤n(n 为正整数)。现要求设计一个时间复杂度尽可能高效的算法,对于链表中 data 的绝对值相等的结点，仅保留第一次出现的结点而删除其余绝对值相等的结点。例如，若给定的单链表 head 如下：



则删除结点后的 head 为：



1) 给出算法的基本设计思想

2) 使用 C/C++语言，给出单链表结点的数据类型定义

3) 根据设计思想，采用 C/C++语言描述算法，关键之处给出注释

4) 说明算法的时间，空间复杂度

参考答案：

(1) 算法的基本设计思想

算法的核心思想是用空间换时间。使用辅助数组记录链表中已出现的数值，从而只需对链表进行一趟扫描。

因为|data|≤n，故辅助数组 q 的大小为 n+1，各元素的初值均为 0。依次扫描链表中的各结点，同时检查 q[|data|]
的值，如果为 0，则保留该结点，并令 q[|data|]=1; 否则，将该结点从链表中删除。

(2) 使用 C 语言描述的单链表结点的数据类型定义

```
typedef struct node {
        int                         data:
        struct node              *link:
}     NODE;
typedefNODE*PNODE;
```

(3) 算法实现

```
void func(PNODE h, int n)
{ PNODE p: h, r;
int*q, m;
q=(int*)malloc(sizeof(int)*(n+1));  //申请 n+1
                                                    //个位置的
                                                    //辅助空间
for(int i=0; i<n+1;i++)                  //数组元素初值置 0
*(q+i)= 0;
while(p->link!=NULL)
{    m=p->link->data>0?p->link->data: -p->link->data;
```

```
    if(*(q+m)==0)                                  //判断该结点的 data 是否已
                                                          //出现过
    {*(q+m)=1;                                     //首次出现
    p=p->link;                                     //保留
    }
    else                                                  //重复出现
    {     r=p->link;                               //删除
    p->link=r->link;
    free(r);
    }
    }
    free(q);
    }
```

2.设计在链式存储结构上交换二叉树中所有结点左右子树的算法。

答案：

```
typedef struct node {int data; struct node *lchild,*rchild;} bitree;
void swapbitree(bitree *bt)
{
   bitree *p;
   if(bt==0) return;
swapbitree(bt->lchild); swapbitree(bt->rchild);
p=bt->lchild; bt->lchild=bt->rchild; bt->rchild=p;
}
```