得分

一、单项选择题（X 题，每题 X 分，共 X 分）

评分标准：每题回答正确得 1.5 分，错误不得分！

1、建立一个长度为 n 的有序单链表的时间复杂度为（ **C** ）。

A. O(n)　　B. O(1)　　C. O(n^2)　　D. O($\log_2 n$)

2、快速排序在最坏情况下的时间复杂度为（ **D** ）。

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

3、对 n 个记录的文件进行快速排序，所需要的辅助存储空间大致为（ **C** ）。

A. O(1)　　B. O(n)　　C. O(1$\log_2$n)　　D. O(n^2)

4、用某种排序方法对关键字序列（25，84，21，47，15，27，68，35，20）进行排序时，序列的变化情况如下，则所采用的排序方法是（ **D** ）。

20，15，21，25，47，27，68，35，84

15，20，21，25，35，27，47，68，84

15，20，21，25，27，35，47，68，84

A. 选择排序　　B. 希尔排序　　C. 归并排序　　D. 快速排序

5、设有以下四种排序方法，则（ **B** ）的空间复杂度最大。

A. 冒泡排序　　B. 快速排序　　C. 堆排序　　D. 希尔排序

6、设一组初始记录关键字序列(5，2，6，3，8)，以第一个记录关键字 5 为基准进行一趟快速排序的结果为（ **C** ）。

A. 2，3，5，8，6　　B. 3，2，5，8，6

C. 3，2，5，6，8　　D. 2，3，6，5，8

7、设有 n 个待排序的记录关键字，则在堆排序中需要（ **A** ）个辅助记录单元。

A. 1　　B. n　　C. n$\log_2$n　　D. n^2

8、设一组初始关键字记录关键字为(20，15，14，18，21，36，40，10)，则以 20 为基准记录的一趟快速排序结束后的结果为（ **A** ）。

A. 10，15，14，18，20，36，40，21　　B. 10，15，14，18，20，40，36，21

C. 10，15，14，20，18，40，36，2l　　D. 15，10，14，18，20，36，40，21

9、设有 5000 个待排序的记录关键字，如果需要用最快的方法选出其中最小的 10 个记录关键字，则用下列（ **B** ）方法可以达到此目的。

A. 快速排序　　B. 堆排序　　C. 归并排序　　D. 插入排序

10、下列四种排序中（ **D** ）的空间复杂度最大。

A. 插入排序　　B. 冒泡排序　　C. 堆排序　　D. 归并排序

11、设一组初始记录关键字序列为(345，253，674，924，627)，则用基数排序需要进行（ **A** ）趟的分配和回收才能使得初始关键字序列变成有序序列。

A. 3　　B. 4　　C. 5　　D. 8

12、下列四种排序中（ **A** ）的空间复杂度最大。

A. 快速排序　　B. 冒泡排序　　C. 希尔排序　　D. 堆

13、设一组初始记录关键字序列为(50，40，95，20，15，70，60，45)，则以增量 d=4 的一趟希尔排序结束后前 4 条记录关键字为（ **B** ）。

A. 40，50，20，95　　B. 15，40，60，20

C. 15，20，40，45　　D. 45，40，15，20

14、设一组初始记录关键字序列为(25，50，15，35，80，85，20，40，36，70)，其中含有 5 个长度为 2 的有序子表，则用归并排序的方法对该记录关键字序列进行一趟归并后的结果为（ **A** ）。

A. 15，25，35，50，20，40，80，85，36，70

B. 15，25，35，50，80，20，85，40，70，36

C. 15，25，35，50，80，85，20，36，40，70

D. 15，25，35，50，80，20，36，40，70，85

15、设一组初始记录关键字序列为(45，80，55，40，42，85)，则以第一个记录关键字 45 为基准而得到一趟快速排序的结果是（ **C** ）。

A. 40，42，45，55，80，83　　B. 42，40，45，80，85，88

C. 42，40，45，55，80，85　　D. 42，40，45，85，55，80

16、执行一趟快速排序能够得到的序列是（ **A** ）。

A. [41，12，34，45，27] 55 [72，63]　　B. [45，34，12，41] 55 [72，63，27]

C. [63，12，34，45，27] 55 [41，72]　　D. [12，27，45，41] 55 [34，63，72]

17、时间复杂度不受数据初始状态影响而恒为 $O(n\log_2 n)$的是（ **A** ）。

A. 堆排序　　B. 冒泡排序　　C. 希尔排序　　D. 快速排序

18、一趟排序结束后不一定能够选出一个元素放在其最终位置上的是（ **D** ）。

A. 堆排序　　B. 冒泡排序　　C. 快速排序　　D. 希尔排序

19、二路归并排序的时间复杂度为（ **C** ）。

A. O(n)　　B. $O(n^2)$　　C. $O(n\log_2 n)$　　D. $O(1og_2 n)$

20、设一组初始记录关键字序列为(60，80，55，40，42，85)，则以第一个关键字 60 为基准而得到的一趟快速排序结果是（ **C** ）。

A. 40，42，60，55，80，85　　B. 42，45，55，60，85，80

C. 42，40，55，60，80，85　　D. 42，40，60，85，55，80

21、利用直接插入排序法的思想建立一个有序线性表的时间复杂度为（ **C** ）。

A. O(n)　　B. $O(n\log_2 n)$　　C. $O(n^2)$　　D. $O(1og_2 n)$

22、下列各种排序算法中平均时间复杂度为 $O(n^2)$是（ **D** ）。

A. 快速排序　　B. 堆排序　　C. 归并排序　　D. 冒泡排序

23、设一组初始记录关键字的长度为 8，则最多经过（ **B** ）趟插入排序可以得到有序序列。

A. 6　　B. 7　　C. 8　　D. 9

24、设一组初始记录关键字序列为(Q，H，C，Y，P，A，M，S，R，D，F，X)，则按字母升序的第一趟冒泡排序结束后的结果是（ **D** ）。

A. F，H，C，D，P，A，M，Q，R，S，Y，X

B. P，A，C，S，Q，D，F，X，R，H，M，Y

C. A，D，C，R，F，Q，M，S，Y，P，H，X

D. H，C，Q，P，A，M，S，R，D，F，X，Y

得分

二、填空题（X 题，每题 X 分，共 X 分）

评分标准：每空回答正确得 1 分，错误不得分，不完全正确则酌情给分！

1、当待排序的记录数较大，排序码较随机且对稳定性不作要求时，宜采用【1】**快速**排序；当待排序的记录数较大，存储空间允许且要求排序是稳定时，宜采用【2】**归并**排序。

2、在堆排序的过程中，对任一分支结点进行筛运算的时间复杂度为【3】**$O(\log_2 n)$**，整个堆排序过程的时间复杂度为【4】**$O(n\log_2 n)$**。

3、在快速排序、堆排序、归并排序中，【5】**归并**排序是稳定的。

4、在堆排序的过程中，对任一分支结点进行筛运算的时间复杂度为【6】**$O(\log_2 n)$**，整个堆排序过程的时间复杂度为【7】**$O(n\log_2 n)$**。

5、在单链表上难以实现的排序方法有【8】**快速排序**、【9】**堆排序**和【10】**希尔排序**。

6、多重表文件和倒排文件都归属于【11】**多关键字**文件。

7、快速排序的最坏时间复杂度为【12】**$O(n^2)$**，平均时间复杂度为【13】**$O(n\log_2 n)$**。

8、设一组初始记录关键字序列为(55，63，44，38，75，80，31，56)，则利用筛选法建立的初始小根堆为 【14】**(31，38，44，56，75，80，55，63)** 。

9、设一组初始记录关键字为(72，73，71，23，94，16，5)，则以记录关键字 72 为基准的一趟快速排序结果为 【15】**(5，16，71，23，72，94，73)** 。

10、设有 n 个无序的记录关键字，则直接插入排序的时间复杂度为 【16】**$O(n^2)$** ，快速排序的平均时间复杂度为 【17】**$O(n\log_2 n)$** 。

11、设初始记录关键字序列为(K1，K2，…，Kn)，则用筛选法思想建堆必须从第 【18】**n/2** 个元素开始进行筛选。

12、设一组初始记录关键字序列为(20，18，22，16，30，19)，则以 20 为中轴的一趟快速排序结果为 【19】**(19，18，16，20，30，22)** 。

13、设一组初始记录关键字序列为(20，18，22，16，30，19)，则根据这些初始关键字序列建成的初始小根堆为 【20】**(16，18，19，20，30，22)** 。

14、设一组初始记录关键字序列(k1，k2，……，kn)是堆，则对 i=1，2，…，n/2 而言满足的条件为 【21】**$k_i<=k_{2i}$ && $k_i<=k_{2i+1}$** 。

15、下面程序段的功能是实现冒泡排序算法，请在下划线处填上正确的语句。

```
void bubble(int    r[n]) {
    for(i=1; i<=n-1; i++) {
        for(exchange=0, j=0; j<  【22】n-i  ; j++)
            if (r[j]>r[j+1]) {
                temp=r[j+1];
                【23】r[j+1]=r[j]  ;
                r[j]=temp;
                exchange=1;
            }
        if(exchange==0) return;
        }
}
```

16、简单选择排序和直接插入排序算法的平均时间复杂度为 【24】**$O(n^2)$** 。

17、快速排序算法的空间复杂度平均情况下为 【25】**$O(\log_2 n)$** ，最坏的情况下为 【26】**O(n)** 。

18、设关键字序列为(Kl，K2，…，Kn)，则用筛选法建初始堆必须从第 【27】**n/2** 个元素开始进行筛选。

19、设有一组初始关键字序列为(24，35，12，27，18，26)，第 1 趟插入执行元素 35 的插入，则第 3 趟直接插入排序结束后的结果的是 【28】**(12，24，27，35，18，26)** 。

20、设有一组初始关键字序列为(24，35，12，27，18，26)，则第 3 趟简单选择排序结束后的

结果的是【29】**(12，18，24，27，35，26)**。

21、下面程序段的功能是实现一趟快速排序，请在下划线处填上正确的语句。

```
struct record {
    int key;
    datatype others;
};

void quickpass(struct record r[], int s, int t, int &i) {
    int j=t;
    struct record x=r[s];
    i=s;
    while(i<j) {
        while (i<j && r[j].key>x.key) j=j-1;
        if (i<j) {
            r[i]=r[j];
            i=i+1;
        }
        while (【30】i<j && r[i].key<x.key ) i=i+1;
        if (i<j) {
            r[j]=r[i];
            j=j-1;
        }
    }
    【31】r[i]=x ;
}
```

22、设一组初始记录关键字序列为(49，38，65，97，76，13，27，50)，则以 d=4 为增量的一趟希尔排序结束后的结果为【32】**(49，13，27，50，76，38，65，97)**。

23、设一组初始记录关键字序列为(49，38，65，97，76，13，27，50)，则第 4 趟直接选择排序结束后的结果为【33】**(13，27，38，49，76，97，65，50)**。

24、设一组初始关键字序列为(38，65，97，76，13，27，10)，则第 3 趟冒泡排序结束后的结果为【34】**(38，13，27，10，65，76，97)**。

25、设一组初始关键字序列为(38，65，97，76，13，27，10)，则第 3 趟简单选择排序后的结果为【35】**(10，13，27，76，65，97，38)**。

26、设有一组初始记录关键字序列为(50，16，23，68，94，70，73)，则将它们调整成初始堆只需把 16 与【36】**50** 相互交换即可。

27、对一组初始关键字序列（40，50，95，20，15，70，60，45，10）进行冒泡排序，则第一趟需要进行相邻记录的比较的次数为【37】**8**，在整个排序过程中最多需要进行

【38】**8** 趟排序才可以完成。

28、在堆排序和快速排序中，如果从平均情况下排序的速度最快的角度来考虑应最好选择【39】**快速** 排序，如果从节省存储空间的角度来考虑则最好选择【40】**堆** 排序。

29、设一组记录关键字序列为(80，70，33，65，24，56，48)，则用筛选法建成的初始堆为【41】**(24，65，33，80，70，56，48)** 。

30、设需要对 5 个不同的记录关键字进行排序，则至少需要比较【42】**4** 次，至多需要比较【43】**10** 次。

31、快速排序算法的平均时间复杂度为【44】**$O(n\log_2 n)$** ，直接插入排序算法的平均时间复杂度为【45】**$O(n^2)$** 。

得分	

三、判断题（X 题，每题 X 分，共 X 分。正确填‘T’，错误“F”。）

评分标准：每题回答正确得 2 分，错误不得分！

1、冒泡排序在初始关键字序列为逆序的情况下执行的交换次数最多。（ **T** ）

2、层次遍历初始堆可以得到一个有序的序列。（ **F** ）

3、快速排序是排序算法中平均性能最好的一种排序。（ **T** ）

4、设某堆中有 n 个结点，则在该堆中插入一个新结点的时间复杂度为 $O(\log_2 n)$。（ **T** ）

5、设初始记录关键字基本有序，则快速排序算法的时间复杂度为 $O(n\log_2 n)$。（ **F** ）

6、希尔排序算法的时间复杂度为 $O(n^2)$。（ **F** ）

7、堆是完全二叉树，完全二叉树不一定是堆。（ **T** ）

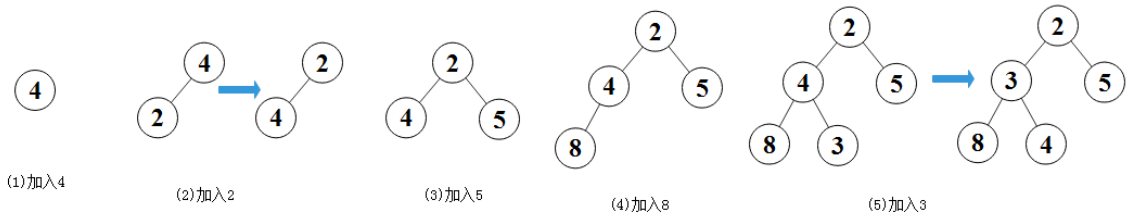
得分	

四、分析题（X 题，每题 X 分，共 X 分）

评分标准：每题回答完全正确得 5 分，其余按得分点给分！

1、画出向小根堆中加入数据 4, 2, 5, 8, 3 时，每加入一个数据后堆的变化。（评分标准：*第 1 点 4 分，第 2 点 1 分！*）

答：



2、已知一组记录的排序码为（46，79，56，38，40，80，95，24），写出对其进行快速排序的每一次划分结果。（评分标准：*第 1 点 4 分，第 2 点 1 分!*）

答：

划分次序	划分结果
第一次	[24 40 38] 46 [56 80 95 79]
第二次	24 [40 38] 46 [56 80 95 79]
第三次	24 [38] 40 46 [56 80 95 79]
第四次	24 38 40 46 56 [80 95 79]
第五次	24 38 40 46 56 [79] 80 [95]
第六次	24 38 40 46 56 79 80 95

注：第六次，相当于【整理】过程，可要可不要。

得分

五、应用题（X 题，每题 X 分，共 X 分）

评分标准：每题回答完全正确得 5 分，其余按得分点给分！

1、已知一维数组中的数据为（18,12,25,53,18），试写出插入排序（升序）过程。并指出具有 n 个元素的插入排序的时间复杂度是多少？（评分标准：*共 5 分。第 1 点 4 分，第 2 点 1 分!*）

答：（1） （4 分）

初始关键字：	[18]	12	25	53	***18***
第一趟：	[12	18]	25	53	***18***
第二趟：	[12	18	25]	53	***18***
第三趟：	[12	18	25	53]	***18***
第四趟：	[12	18	***18***	25	53]

（2）$O(n^2)$ （1 分）；

2、设一组初始记录关键字序列为(19，21，16，5，18，23)，要求给出以 19 为基准的一趟快速排序结果以及第 2 趟直接选择排序后的结果。（评分标准：*共 5 分。第 1 点 4 分，第 2 点 1 分!*）

答：（1）(18,5,16,19,21,23)。

（2）(5，16，21，19，18，23)；

3、设一组初始记录关键字序列为(45，80，48，40，22，78)，则分别给出第 4 趟简单选择排序和第 4 趟直接插入排序后的结果。（评分标准：*共 5 分。第 1 点 4 分，第 2 点 1 分!*）

答：（1）(22，40，45，48，80，78)；

（2）(22，40，45，48，80， 78)；

得分

六、编程题（X 题，每题 X 分，共 X 分）

评分标准：每小题回答正确得 10 分，不完全正确则按得分点给分。

1、阅读下列函数 arrange( )。（1）写出该函数的功能；（2）写一个调用上述函数实现下列功能的算法：对一整型数组 b[n]中的元素进行重新排列，将所有负数均调整到数组的低下标端，

将所有正数均调整到数组的高下标端，若有零值，则置于两者之间，并返回数组中零元素的个数。实现算法的函数原型为：int ***f***(int b[], int n);（评分标准：*第1点4分，第2点1分!*）

```
int arrange(int a[], int 1, int h, int x) {   //1 和 h 分别为数据区的下界和上界
    int i,j,t;
    i=1;
    j=h;
    while(i<j) {
        while(i<j && a[j]>=x)
            j--;
        while(i<j && a[j]>=x)
            i++;
        if(i<j) {
            t = a[j];
            a[j] = a[i];
            a[i] = t;
        }
    }

    if(a[i]<x)
        return i;
    else
        return i-1;
}
```

答：（1）该函数的功能是：调整整数数组 a[]中的元素并返回分界值 i，使所有＜x 的元素均落在 a[1..i]上，使所有≥x 的元素均落在 a[i＋1..h]上。

```
int f(int b[], int n) {
    int p,q;

    p=arrange(b,0,n-1,0);

    q=arrange(b,p+1,n-1,1);

    return q-p;
}
```

或

```
int f(int b[], int n) {
    int p,q;
    p=arrange(b,0,n-1,1);
    q=arrange(b,0,p,0);

    return p-q;
}
```

2、设有一组初始记录关键字序列（K_1，K_2，…，K_n），要求设计一个算法能够在 O(n)的时间复杂度内将线性表划分成两部分，其中左半部分的每个关键字均小于 K_i，右半部分的每个关键字均大于等于 K_i。实现算法的函数原型为：void ***quickpass***(int r[], int s, int t);（评分标准：*第1点4分，第2点1分!*）

答：

```
void quickpass(int r[], int s, int t) {
    int i=s, j=t, x=r[s];
```

```
    while(i<j) {
        while(i<j && r[j]>x)
            j=j-1;

        if (i<j) {
            r[i]=r[j];
            i=i+1;
        }

        while(i<j && r[i]<x)
            i=i+1;

        if (i<j) {
            r[j]=r[i];
            j=j-1;
        }
    }
    r[i]=x;
}
```

3、设计将所有奇数移到所有偶数之前的算法。实现算法的函数原型为：void ***quickpass***(int r[], int s, int t)；（评分标准：*第 1 点 4 分，第 2 点 1 分!*）

答：

```
void quickpass(int r[], int s, int t) {
    int i=s,j=t,x=r[s];
    while(i<j) {
        while(i<j && r[j]%2==0)
            j=j-1;
        if (i<j) {
            r[i]=r[j];
            i=i+1;
        }

        while(i<j && r[i]%2==1)
            i=i+1;
        if (i<j) {
            r[j]=r[i];
            j=j-1;
        }
    }
    r[i]=x;
}
```

4、设计两个有序单链表的合并排序算法。实现算法的函数原型为：void ***mergelklist***(lklist *ha, lklist *hb, lklist *hc)；（评分标准：*第1点4分，第2点1分!*）

答：
```
void mergelklist(lklist *ha, lklist *hb, lklist *hc) {
    lklist *s=hc=0;
    while(ha!=0 && hb!=0)
        if(ha->data<hb->data) {
            if(s==0)
                hc=s=ha;
            else {
                s->next=ha;
                s=ha;
            }

            ha=ha->next;
        } else {
            if(s==0)
                hc=s=hb;
            else {
                s->next=hb;
                s=hb;
            }

            hb=hb->next;
        }

    if(ha==0)
        s->next=hb;
    else
        s->next=ha;
}
```

5、在链式存储结构上设计直接插入排序算法。实现算法的函数原型为：void ***straightinsertsort***(lklist *&head)；（评分标准：*第1点4分，第2点1分!*）

答：
```
void straightinsertsort(lklist *&head) {
    lklist *s,*p,*q;
    int t;
    if (head==0 || head->next==0)
        return;
    else
        for(q=head,p=head->next; p!=0; p=q->next) {
            for(s=head; s!=q->next; s=s->next)
```

```
                if (s->data>p->data)
                    break;

            if(s==q->next)
                q=p;
            else {
                q->next=p->next;
                p->next=s->next;
                s->next=p;
                t=p->data;
                p->data=s->data;
                s->data=t;
            }
        }
}
```

6、设计在链式结构上实现简单选择排序算法。实现算法的函数原型为：void ***simpleselectsorlklist***(lklist *&head);（评分标准：*第1点4分，第2点1分!*）

答：

```
void simpleselectsorlklist(lklist *&head) {
    lklist *p,*q,*s;
    int min,t;
    if(head==0 ||head->next==0)
        return;

    for(q=head; q!=0; q=q->next) {
        min=q->data;
        s=q;
        for(p=q->next; p!=0; p=p->next)
            if(min>p->data) {
                min=p->data;
                s=p;
            }

        if(s!=q) {
            t=s->data;
            s->data=q->data;
            q->data=t;
        }
    }
}
```

7、设计在链式存储结构上合并排序的算法。实现算法的函数原型为：void ***mergelklist***(lklist *ha,

lklist *hb, lklist *&hc)；（评分标准：第1点4分，第2点1分！）

答：
```
void mergelklist(lklist *ha, lklist *hb, lklist *&hc) {
    lklist *s=hc=0;
    while(ha!=0 && hb!=0)
        if(ha->data<hb->data) {
            if(s==0)
                hc=s=ha;
            else {
                s->next=ha;
                s=ha;
            }

            ha=ha->next;
        } else {
            if(s==0)
                hc=s=hb;
            else {
                s->next=hb;
                s=hb;
            }

            hb=hb->next;
        }

    if(ha==0)
        s->next=hb;
    else
        s->next=ha;
}
```

8、设关键字序列(k_1，k_2，…，k_{n-1})是堆，设计算法将关键字序列(k_1，k_2，…，k_{n-1}，x)调整为堆。

实现算法的函数原型为：void ***adjustheap***(int r[ ], int n)；（评分标准：第1点4分，第2点1分！）

答：
```
void adjustheap(int r[ ], int n) {
    int j=n,i=j/2,temp=r[j-1];
    while (i>=1)
        if (temp>=r[i-1])
            break;
        else {
            r[j-1]=r[i-1];
            j=i;
            i=i/2;
```

```
            }

        r[j-1]=temp;
}
```