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、折半查找法适用于（ **A** ）。

A. 有序顺序表　　B. 有序单链表

C. 有序顺序表和有序单链表都可以　　D. 无限制

2、采用开放定址法处理散列表的冲突时，其平均查找长度（ **B** ）。

A. 低于链接法处理冲突　　B. 高于链接法处理冲突

C. 与链接法处理冲突相同　　D. 高于二分查找

3、从二叉搜索树中查找一个元素时，其时间复杂度大致为（ **C** ）。

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

4、若有 18 个元素的有序表存放在一维数组 A[0..18]中，第一个元素放在 A[1]中，现进行二分查找，则查找 A[3]的比较序列的下标依次为（ **D** ）。

A. 1，2，3　　B. 9，5，2，3　　C. 9，5，3　　D. 9，4，2，3

5、若有 18 个元素的有序表存放在一维数组 A[0..17]中，第一个元素放在 A[0]中，现进行二分查找，则查找 A[3]的比较序列的下标依次为（ **B** ）。

A. 1，2，3　　B. 8，3　　C. 8，4，3　　D. 9，4，2，3

6、对于线性表（7，34，55，25，64，46，20，10）进行散列存储时，若选用 H(K)=K%9 作为散列函数，则散列地址为 1 的元素有（ **D** ）个。

A. 1　　B. 2　　C. 3　　D. 4

7、在一个长度为 n 的顺序线性表中顺序查找值为 x 的元素时，查找成功时的平均查找长度（即 x 与元素的平均比较次数，假定查找每个元素的概率都相等）为（ **C** ）。

A. n　　B. n/2　　C. (n+1)/2　　D. (n-1)/2

8、适于对动态查找表进行高效率查找的组织结构是（ **C** ）。

A. 有序表　　B. 分块有序表　　C. 三叉排序树　　D. 线性链表

9、不定长文件是指（ **B** ）。

A. 文件的长度不固定　　B. 记录的长度不固定

C. 字段的长度不固定　　D. 关键字项的长度不固定

10、设二叉排序树中有 n 个结点，则在二叉排序树的平均查找长度为（ **B** ）。

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

11、在二叉排序树中插入一个结点的平均时间复杂度为（ **C** ）。

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

注：最差情况下是 O(n)，如果是最一般最基础的二叉树的话，因为深度不平衡，所以会发展成单链的形状,就是一条线 n 个点那么深。如果是深度平衡的二叉树 O($\log_2 n$)！
不是问平均时间复杂度，就是 O(n)；问平均时间复杂度，是 O($\log_2 n$)！

12、在二叉排序树中插入一个结点最坏情况下的时间复杂度为（ **B** ）。

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

13、设有序顺序表中有 n 个数据元素，则利用二分查找法查找数据元素 X 的最多比较次数不超过（ **A** ）。

A. $\log_2 n+1$　　B. $\log_2 n-1$　　C. $\log_2 n$　　D. $\log_2(n+1)$

14、设有序表中有 1000 个元素，则用二分查找查找元素 X 最多需要比较（ **B** ）次。

A. 25　　B. 10　　C. 7　　D. 1

15、顺序查找不论在顺序线性表中还是在链式线性表中的时间复杂度为（ **A** ）。

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

16、设二叉排序树上有 n 个结点，则在二叉排序树上查找结点的平均时间复杂度为（ **D** ）。

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

17、（ **B** ）二叉排序树可以得到一个从小到大的有序序列。

A. 先序遍历　　B. 中序遍历　　C. 后序遍历　　D. 层次遍历

18、设一组初始记录关键字序列为(13，18，24，35，47，50，62，83，90，115，134)，则利用二分法查找关键字 90 需要比较的关键字个数为（ **B** ）。

A. 1　　B. 2　　C. 3　　D. 4

19、设某散列表的长度为 100，散列函数 H(k)=k % P，则 P 通常情况下最好选择（ **B** ）。

A. 99　　B. 97　　C. 91　　D. 93

20、在二叉排序树中插入一个关键字值的平均时间复杂度为（ **B** ）。

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

21、设一个顺序有序表 A[1:14]中有 14 个元素，则采用二分法查找元素 A[4]的过程中比较元素的顺序为（ **C** ）。

A. A[1]，A[2]，A[3]，A[4]　　B. A[1]，A[14]，A[7]，A[4]

C. A[7]，A[3]，A[5]，A[4]　　D. A[7]，A[5] ，A[3]，A[4]

22、设散列表中有 m 个存储单元，散列函数 H(key)= key % p，则 p 最好选择（ **B** ）。

A. 小于等于 m 的最大奇数　　B. 小于等于 m 的最大素数

C. 小于等于 m 的最大偶数　　D. 小于等于 m 的最大合数

23、设顺序表的长度为 n，则顺序查找的平均比较次数为（ **C** ）。

A. n　　B. n/2　　C. (n+1)/2　　D. (n-1)/2

24、设有序表中的元素为(13，18，24，35，47，50，62)，则在其中利用二分法查找值为 24 的元素需要经过（ **C** ）次比较。

A. 1　　B. 2　　C. 3　　D. 4

25、设顺序线性表的长度为 30，分成 5 块，每块 6 个元素，如果采用分块查找，则其平均查找长度为（ **D** ）。

A. 6　　B. 11　　C. 5　　D. 6.5

26、设有一组初始记录关键字序列为(34，76，45，18，26，54，92)，则由这组记录关键字生成的二叉排序树的深度为（ **A** ）。

A. 4　　B. 5　　C. 6　　D. 7

27、二叉排序树中左子树上所有结点的值均（ **A** ）根结点的值。

A. <　　B. >　　C. =　　D. !=

28、设有 n 个关键字具有相同的 Hash 函数值，则用线性探测法把这 n 个关键字映射到 HASH 表中需要做（ **C** ）次线性探测。

A. n^2　　B. n(n+1)　　C. n(n+1)/2　　D. n(n-1)/2

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、在线性表的散列存储中，处理冲突的常用方法有 【1】开放定址法 和 【2】链接法 两种。

2、假定一个线性表为(12,23,74,55,63,40)，若按 Key % 4 条件进行划分，使得同一余数的元素成为一个子表，则得到的四个子表分别为 【3】（12，40） 、 【4】（ ） 、 【5】（74） 和 【6】（23, 55, 63） 。

3、向一棵 B_树插入元素的过程中，若最终引起树根结点的分裂，则新树比原树的高度 【7】增加 1 。

4、一棵高度为 5 的理想平衡树中，最少含有 【8】16 个结点，最多含有 【9】31 个结点。

5、在一个索引文件的索引表中，每个索引项包含对应记录的 【10】索引值域 和 【11】开始位置域 两项数据。

6、在有序表（12，24，36，48，60，72，84）中二分查找关键字 72 时所需进行的关键字比较次数为【12】2。

7、设线性表中有 n 个数据元素，则在顺序存储结构上实现顺序查找的平均时间复杂度为【13】O(n)，在链式存储结构上实现顺序查找的平均时间复杂度为【14】O(n)。

8、为了能有效地应用 HASH 查找技术，必须解决的两个问题是【15】构造一个好的 HASH 函数和【16】确定解决冲突的方法。

9、【17】中序遍历二叉排序树中的结点可以得到一个递增的关键字序列（填先序、中序或后序）。

10、设查找表中有 100 个元素，如果用二分法查找方法查找数据元素 X，则最多需要比较【18】7次就可以断定数据元素 X 是否在查找表中。

11、下列算法实现在顺序散列表中查找值为 x 的关键字，请在下划线处填上正确的语句。

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

int hashsqsearch(struct record hashtable[ ], int k) {
    int i,j;
    j=i=k % p;
    while(hashtable[j].key!=k&&hashtable[j].flag!=0) {
        j=(【19】j+1) %m;
        if (i==j) return(-1);
    }

    if (【20】hashtable[j].key==k) return(j);
    else return(-1);
}
```

12、下列算法实现在二叉排序树上查找关键值 k，请在下划线处填上正确的语句。

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

bitree *bstsearch(bitree *t, int  k) {
    if (t==0 ) return(0);
    else
        while (t!=0)
```

```
        if (t->key==k)  【21】return (t)  ;
        else if (t->key>k) t=t->lchild;
        else  【22】t=t->rchild  ;
}
```

13、根据初始关键字序列(19，22，01，38，10)建立的二叉排序树的高度为【23】3 。

14．设散列函数 H(k)=k mod p，解决冲突的方法为链地址法。要求在下列算法划线处填上正确的语句完成在散列表 hashtalbe 中查找关键字值等于 k 的结点，成功时返回指向关键字的指针，不成功时返回标志 0。

```
typedef struct node {
    int key;
    struct node *next;
} lklist;

void createlkhash(lklist *hashtable[ ]) {
    int i,k;
    lklist *s;
    for(i=0; i<m; i++)  【24】hashtable[i]=0  ;
    for(i=0; i<n; i++) {
        s=(lklist *)malloc(sizeof(lklist));
        s->key=a[i];
        k=a[i] % p;
        s->next=hashtable[k];
        【25】hashtable[k]=s  ;
    }
}
```

14、下面程序段的功能是实现二分查找算法，请在下划线处填上正确的语句。

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

int bisearch(struct record r[ ], int k) {
    int low=0,mid,high=n-1;
    while(low<=high) {
        【26】mid=(low+high)/2  ;
        if(r[mid].key==k) return(mid+1);
        else if(【27】r[mid].key>k  ) high=mid-1;
        else low=mid+1;
    }
    return(0);
```

}

15、散列表中解决冲突的两种方法是【28】开放定址法和【29】链地址法。

16、解决散列表冲突的两种方法是【30】开放定址法和【31】链地址法。

17、下面程序段的功能是实现在二叉排序树中插入一个新结点，请在下划线处填上正确的内容。

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

void   bstinsert(bitree *&t,int k) {
    if (t==0 ) {
        【32】t=(bitree *)malloc(sizeof(bitree));
        t->data=k;
        t->lchild=t->rchild=0;
    } else if (t->data>k) bstinsert(t->lchild,k);
    else 【33】bstinsert(t->rchild, k);
}
```

18、设一组初始记录关键字序列为(20，12，42，31，18，14，28)，则根据这些记录关键字构造的二叉排序树的平均查找长度是【34】19/7。

19、设二叉排序树的高度为 h，则在该树中查找关键字 key 最多需要比较【35】h 次。

20、设在长度为 20 的有序表中进行二分查找，则比较一次查找成功的结点数有【36】1 个，比较两次查找成功有结点数有【37】2 个。

21、设散列表的长度为 8，散列函数 H(k)=k % 7，用线性探测法解决冲突，则根据一组初始关键字序列(8，15，16，22，30，32)构造出的散列表的平均查找长度是【38】8/3。

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、分块查找的平均查找长度不仅与索引表的长度有关，而且与块的长度有关。（ T ）

2、当向二叉排序树中插入一个结点，则该结点一定成为叶子结点。（ T ）

3、如果两个关键字的值不等但哈希函数值相等，则称这两个关键字为同义词。（ T ）

4、分块查找的基本思想是首先在索引表中进行查找，以便确定给定的关键字可能存在的块号，然后再在相应的块内进行顺序查找。（ T ）

5、向二叉排序树中插入一个结点需要比较的次数可能大于该二叉树的高度。　　（ F ）

6、顺序表查找指的是在顺序存储结构上进行查找。　　（ F ）

得分

四、程序填空题（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、如下为二分查找的非递归算法，请在下划线处填上正确的语句。（评分标准：*第 1 点 3 分！第 2 点 2 分！*）

```
int Binsch(ElemType A[ ],int n,KeyType K) {
    int low=0;
    int high=n-1;
    while (low<=high) {
        int mid=____【1】(low+high)/2____;
        if(K==A[mid].key)    return mid;//查找成功，返回元素的下标
        else if(K<[mid].key)
            ____【2】high=mid-1____; //在左子表上继续查找
        else  ____【3】low=mid+1____; //在右子表上继续查找
    }
    return -1;    //查找失败，返回-1
}
```

2、以下程序的功能是二叉搜索树的查找——递归算法，请在下划线处填上正确的语句。（评分标准：*第 1 点 3 分！第 2 点 2 分！*）

```
bool Find(BTreeNode* BST, ElemType& item) {
    if(BST==NULL)
        return false; //查找失败
    else {
        if(item==BST->data) {
            item=BST->data; //查找成功
            return  ____【4】true____;
        } else if(item<BST->data) {
            return    Find(____【5】BST->left____, item);
        } else {
            return Find(____【6】BST->right____, item);
        }
    }
}
```

3、以下代码实现对顺序存储的有序表进行二分查找的递归算法，请在下划线处填上正确的语句。（评分标准：*第 1 点 3 分！第 2 点 2 分！*）

```
int Binsch( ElemType A[ ], int low, int high, KeyType K ) {
    if (low <= high) {
        int mid = 【7】(low + high)/2 ;
        if ( K= = A[ mid ].key )
            return mid;
        else if ( K < A[mid].key)
            return 【8】Binsch(A, low, mid–1, K) ;
        else
            return 【9】Binsch(A, mid+1, high, K); ;
    } else {
        return 【10】-1 ;
    }
}
```

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、设散列函数 H(k)=k % 13，设关键字系列为{22, 12, 24, 6, 45, 7, 8, 13, 21}，要求用线性探测法处理冲突。（评分标准：*共 6 分。第 1 点 4 分，第 2 点 1 分!*）

（1）构造 HASH 表。

（2）分别求查找成功和不成功时的平均查找长度。

答：（1）Hash 表（4 分）。

地址	0	1	2	3	4	5	6	7	8	9	10	11	12
关键安	13	21					6	45	7	22	8	24	12
探测次数	1	7					1	2	2	1	3	1	1

（2）查找成功时的平均查找长度：（5*1 + 2*2 + 1*3 + 1*7）/9=19/9　　（1 分）；

查找不成功时的平均查找长度：（【3+2】+【1+1+1+1】+ 【10+9+8+7+6+5+4】/13=58/13　（1 分）

注：找到2,3,4,5不成功的查找次数=1，找到6不成功的查找次数=10（找到6不成功, 根据冲突处理办法，要探测7,8,9,10, 11, 12, 0, 1, 2。直到2时探测失败，查找不成功），……

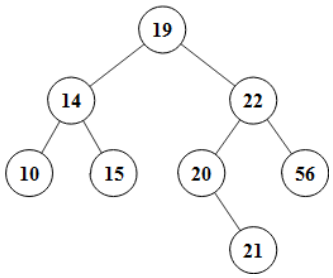
2、给定表（19,14,22,15,20,21,56,10）。（评分标准：*共 8 分。第 1 点 4 分，第 2 点 1 分!*）

（1）按元素在表中的次序，建立一棵二叉排序树

（2）对(1)中所建立的二叉排序树进行中序遍历，写出遍历序列。

（3）画出对(2)中的遍历序列进行折半查找过程的判定树。

答：（1）构造的二叉树：　　（3 分）



（2）10 14 15 19 20 21 22 56。　　（2 分）

（3）；

3、设一组初始记录关键字集合为(25，10，8，27，32，68)，散列表的长度为 8，散列函数 H(k)=k mod 7，要求分别用线性探测和链地址法作为解决冲突的方法设计哈希表。（评分标准：*共5分。第1点4分，第2点1分!*）

0	1	2	3	4	5	6	7
Λ	8	Λ	10	25	32	27	68

答：（1）线性探测：。

h_0

$h_1 -> 8$

h_2

$h_3 -> 10$

$h_4 -> 25 -> 32$

$h_5 -> 68$

（2） $h_6 -> 27$ ；

4、设一组有序的记录关键字序列为(13，18，24，35，47，50，62，83，90)，查找方法用二分查找，要求计算出查找关键字 62 时的比较次数并计算出查找成功时的平均查找长度。（评分标准：*共5分。第1点4分，第2点1分!*）

答：（1）2；

（2）ASL=91*1+2*2+3*4+4*2)=25/9；

5、设有一组初始记录关键字为(45，80，48，40，22，78)，要求构造一棵二叉排序树并给出构造过程。（评分标准：*共5分。第1点4分，第2点1分!*）

答：略。

6、设一组初始记录关键字序列为(15，17，18，22，35，51，60)，要求计算出成功查找时的平均查找长度。（评分标准：*共5分。第1点4分，第2点1分!*）

答：ASL=(1*1+2*2+3*4)/7=17/7；

7、设散列表的长度为 8，散列函数 H(k)=k mod 7，初始记录关键字序列为(25，31，8，27，13，68)，要求分别计算出用线性探测法和链地址法作为解决冲突方法的平均查找长度。（评分标准：*共5分。第1点4分，第2点1分!*）

答：（1）$ASL_{1=}7/6$

地址	0	1	2	3	4	5	6	7
关键字		8		31	25	68	27	13
探测次数		1		1	1	1	1	2

（2）$ASL_2=7/6$

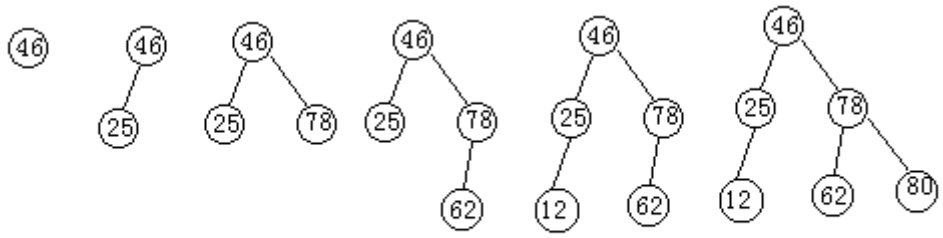
地址	关键字		
0			
1	8		
2			
3	31		
4	25		
5	68		
6	27	-->	13
7			

得分

六、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、设有一个输入数据的序列是{46, 25, 78, 62, 12, 80}，试画出从空树起，逐个输入各个数据而生成的二叉搜索树。（评分标准：*第1 点4 分，第2 点1 分!*）



(a) (b) (c) (d) (e) (f)

答：

2、一个线性表为 B=（12，23，45，57，20，03，78，31，15，36），设散列表为 HT[0..12]，散列函数为 H（key）= key % 13 并用线性探查法解决冲突，请画出散列表，并计算等概率情况下查找成功的平均查找长度。（评分标准：*第1 点4 分，第2 点1 分!*）

答：（1）

0	1	2	3	4	5	6	7	8	9	10	11	12
78		15	03		57	45	20	31		23	36	12

（2）查找成功的平均查找长度：$ASL_{SUCC}=14/10=1.4$

得分

七、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、设计判断一棵二叉树是否是二叉排序树的算法。实现算法的函数原型为：void ***inorder***(bitree *bt)；（评分标准：*第1点4分，第2点1分!*）

答：
```
int minnum=-32768, flag=1;
typedef struct node {
    int key;
    struct node *lchild,*rchild;
} bitree;

void inorder(bitree *bt) {
    if (bt!=0) {
        inorder(bt->lchild);

        if(minnum>bt->key)
            flag=0;

        minnum=bt->key;

        inorder(bt->rchild);
    }
}
```

2、在链式存储结构上建立一棵二叉排序树。实现算法的函数原型为：void ***createbsttree***(bitree *&bt)；（评分标准：*第1点4分，第2点1分!*）

答：
```
#define n 10
typedef struct node {
    int key;
    struct node *lchild,*rchild;
} bitree;

void bstinsert(bitree *&bt,int key) {
    if(bt==0){
        bt=(bitree *)malloc(sizeof(bitree));
        bt->key=key;
        bt->lchild=bt->rchild=0;
    }else if (bt->key>key){
        bstinsert(bt->lchild,key);
    }else{
        bstinsert(bt->rchild,key);
    }
}
```

```
void createbsttree(bitree *&bt) {
    int i;
    for(i=1; i<=n; i++)
        bstinsert(bt,random(100));
}
```

3、设计在顺序有序表中实现二分查找的算法。实现算法的函数原型为：int ***bisearch***(struct record r[ ], int k)；（评分标准：*第1点4分，第2点1分!*）

答：

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

int bisearch(struct record r[ ], int k) {
    int low=0,mid,high=n-1;
    while(low<=high) {
        mid=(low+high)/2;

        if(r[mid].key==k)
            return(mid+1);
        else if(r[mid].key>k)
            high=mid-1;
        else
            low=mid+1;
    }
    return(0);
}
```

4、设计求结点在二叉排序树中层次的算法。实现算法的函数原型为：void ***level***(bitree *bt,int x)；（评分标准：*第1点4分，第2点1分!*）

答：

```
int lev=0;
typedef struct node {
    int key;
    struct node *lchild,*rchild;
} bitree;

void level(bitree *bt,int x) {
    if (bt!=0) {
        lev++;

        if (bt->key==x)
            return;
```

```
        else if (bt->key>x)
            level(bt->lchild,x);
        else
            level(bt->rchild,x);
    }
}
```

5、设计在二叉排序树上查找结点 X 的算法。实现算法的函数原型为：bitree ***bstsearch1***(bitree *t, int key)；（评分标准：*第 1 点 4 分，第 2 点 1 分！*）

答：

```
bitree *bstsearch1(bitree *t, int key) {
    bitree *p=t;
    while(p!=0)
        if (p->key==key)
            return(p);
        else if (p->key>key)
            p=p->lchild;
        else
            p=p->rchild;

    return(0);
}
```