《数据结构》参考答案

一、选择题**（每小题2分，共20分）**

1-5 CBDBC

6-10 ABBCA

二、填空题**（每小题2分，共20分）**

1、10

2、n2-2e

3、 S2+S3

4、开放定址法，链地址法

5、14

6、2h-1，2h-1

7、(12，24，35，27，18，26)

8、(12，18，24，27，35，26)

9、三元组 十字链表

10、0

三、名词解释题**（每小题5分，共15分）**

1. 数据结构：相互之间存在一种或多种特定关系的数据元素的集合，是计算机存储和数据组织的方式，它分为三个方面，即数据的逻辑结构，数据的物理结构，数据的操作．

2. 平衡二叉排序树（AVL树）：或者是空树，或者是符合一下性质的二又排序树

1．左子树和右子树的高度之差的绝对值小于等于1

2左子树和右子树也是平衡二叉排序树

3. 哈夫曼树：又称最优二叉树，是一类带权路径长度最短的树

哈夫曼编码：在哈夫曼树中，约定左分支代表0，右分支代表1，把叶子结点到根结点的路径上的左右分支代表的码从下至上一次连接起来，组成的字符串称为该叶子结点的哈夫曼编码，这就是哈夫曼编码

四、应用题**（每小题6分，共30分）**

**1.** H(30)=30 Mod 7=2 H(15)=15 Mod 7=1 H(21)=21 Mod 7=0

H(40)=40 Mod 7=5 H(25)=25 Mod 7=4 H(26)=26 Mod 7=5（冲突）

H(36)=36 Mod 7=1（冲突） H(37)=37 Mod 7=2（冲突）

（1）把关键字依次填入下表，并统计每个关键字查找成功时比较次数（8分）

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 地址 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 关键字 | 21 | 15 | 30 | 36 | 25 | 40 | 26 | 37 |  |  |
| 比较  次数 | 1 | 1 | 1 | 3 | 1 | 1 | 2 | 6 |  |  |

（2）查找成功时ASL： （2分）

ASL成功=（1+1+1+3+1+1+2+6）/8=2。

2. 答案：1，5，2，3，6，4 1，5，6，2，3，4 5，1，2，3，6，4

5，1，6，2，3，4 5，6，1，2，3，4 1，5，2，6，3，4

5，1，2，6，3，4

3. 

WPL=（1+2）×4+3×3+（4+5+6）×2=51

4.



5.

初始序列：30，10，50，40，45，85，80

第一趟排序：[10] 30 [ 50 40 45 85 80]

第二趟排序：10 30 [45 40] 50 [85 80]

第三趟排序：10 30 40 45 50 [85 80]

第四趟排序：10 30 40 45 50 [80] 85

五、设计题**（每小题3分，共15分）**

1. void reverse(int nums[],int i,int j){

int temp;

while(i<j){

temp=nums[i];

nums[i]=nums[j];

nums[j]=temp;

i++;

j--;

}

}

/\* 将B中的所有元素移动到A的前面 \*/

void move(int A[],int m,int n){

reverse(A,0,m-1);

reverse(A,m,m+n-1);

reverse(A,0,m+n-1);

}

2. void SearchExchangeInsert(SqList &L,ElemType x){

int low = 0,high = L.length - 1,mid;

while(low <= high){

mid = (mid + high)/2;

if(L.data[mid] == x)

break;

else if(L.data[mid] < x)

low = mid + 1;

else

high = mid - 1;

}

if(L.data[mid] == x && mid != L.length-1){

int temp = L.data[mid];

L.data[mid] = L.data[mid + 1];

L.data[mid + 1] = temp;

}

if(high < low)

ListInsert\_Sq(L,low + 1,x);

}

3. int depth(bintree t,char x)

{

int num1,num2,n;//num1,num2分别记录在左子树，右子树中查找到x的层数,n记录最终返回的结果层数

if(t==NULL)

{

return 0;

}

else

{

if(t->data==x)

{

return 1;

}

num1=depth(t->lchild,x);

num2=depth(t->rchild,x);

n=num1+num2; //num1和num2之中必有一个为0

if(num1!=0||num2!=0) //找到了x ，往回数

{

n++;

}

}

return n;

}