《数据结构》参考答案

一、选择题**（每小题2分，共20分）**

1-5 CBCBB

6-10 CBCAA

二、填空题**（每小题2分，共20分）**

1、 (49，13，27，50，76，38，65，97)

2、 （（））

3、 p->next=s

4、head->rlink，p->llink

5、CABD

6、1，16

7、0

8、(13，27，38，50，76，49，65，97)

9、n-1

10、50

三、名词解释题**（每小题5分，共15分）**

1. 线性表：是最常用，最简单的一种数据结构，一个线性表是n个数据元素的有限序列，除首尾元素外，每个元素有唯一的前驱和唯一的后继。

顺序表：采用顺序存储结构的线性表通常称为顺序表。

链表：采用链式存储结构的线性表通常称为顺序表

2. 广义表：是由零个或多个单元素或子表所构成的有限序列，是线性表的推广，也有人称其为列表

3. 矩阵的压缩存储：有的矩阵中有许多值相同元素或者是零元素，为了节省存储空间对这类矩阵采用多个值相同的元素只分配一个存储空间，有时零元素不存储的存储策略，称为矩阵的压缩存储

**四、应用题（每小题6分，共30分）**

**1.** H(9)=9 Mod 7=2 H(1)=1 Mod 7=1 H(23)=23 Mod 7=2（冲突），H1=（2+11）Mod 10=3

H(14)=14 Mod 7=0 H(55)=55 Mod 7=6 H(20)=20 Mod 7=6（冲突），H1=（6+11） Mod 10=7 H(84)=84 Mod 7=0（冲突），H1=（0+11） Mod 10=1（冲突），H2=（0+22）Mod 10=4

H(27)=27 Mod 7=6（冲突），H1=（6+11） Mod 10=7（冲突），H2=（6+22）Mod 10=0（冲突），H3=（6+32）Mod 10=5

（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 | 14 | 1 | 9 | 23 | 84 | 27 | 55 | 20 |  |  |
| 比较次数 | 1 | 1 | 1 | 2 | 3 | 4 | 1 | 2 |  |  |

（2）查找成功时ASL： （2分）

ASL成功=（1+1+1+2+3+4+1+2）/8=15/8。

**2.** (1)图形态： (2)prim算法求最小生成树：

|  |  |
| --- | --- |
|  |  |

3. 构造过程：



WPL：2\*4+3\*4+5\*3+6\*3+7\*3+8\*2+9\*2=108

4.



5. 初始序列 [16] ， 15， 18， 16， 17， 18， 20， 13

第 1 次排序 [15 ， 16] ， 18， 16， 17， 18， 20， 13

第 2 次排序 [15 ， 16， 18] ， 16， 17， 18， 20， 13

第 3 次排序 [15 ， 16， 16， 18] ， 17， 18， 20， 13

第 4 次排序 [15 ， 16， 16， 17， 18] ， 18， 20， 13

第 5 次排序 [15 ， 16， 16， 17， 18， 18] ， 20， 13

第 6 次排序 [15 ， 16， 16， 17， 18， 18， 20] ， 13

第 7 次排序 [13 ， 15， 16， 16， 17， 18， 18， 20]

**五、设计题（每小题3分，共15分）**

1. typedef struct node {

int data;

struct node \*next;

}lklist;

void intersection(lklist \*ha,lklist \*hb,lklist \*&hc)

{

lklist \*p,\*q,\*t;

for(p=ha,hc=NULl;p!=NULL;p=p->next)

{ for(q=hb;q!=NULL;q=q->next) if (q->data==p->data) break;

if(q!=0){ t=(lklist \*)malloc(sizeof(lklist)); t->data=p->data;t->next=hc; hc=t;}

}

}

2.

void Delete\_list(Lnode \*head, ElemType x, ElemType y)

{Lnode \*p, \*q;

if(!head) return ERROR;

p=head; q=p;

while(!p)

{if(p->data>x) && (p->data<y)}i++;

if(p==head)

{head=p->next; free(p);

p=head; q=p; }

else

{q->next=p->next; free(p);

p=q->next; }

else

{q=p; p=p->next; }

}

}

3. TreeNode\* lowestCommonAncestor(TreeNode\* root, TreeNode\* p, TreeNode\* q) {

if (root == NULL || p == root || q == root)

return root;

TreeNode\* left = lowestCommonAncestor(root->left, p, q);

TreeNode\* right = lowestCommonAncestor(root->right, p, q);

if (left != NULL && right != NULL)

return root;

return left != NULL ? left : right;

}