2006 年数据结构参考答案

一、

```
void c1_1(Linklist A, Linklist B, Linklist &C)
{
    LNode *pa=A->next, *pb=B->next,*pc,*s,*r;
    C=( LNode*)malloc(sizeof(LNode));
    C->next=null;
    r=C;
    while(pa)
    {
        s=( LNode*)malloc(sizeof(LNode));
        s->data=pa->data;s->next=null;
        r->next=s;r=s;
        pa=pa->next;
    }
    while(pb)
    {
        pc=C->next;
        while(pc&&pb->data!=pc->data)
            pc=pc->next;
        if(pc==null)
        {
            s=( LNode*)malloc(sizeof(LNode));
            s->data=pb->data;s->next=null;
            r->next=s;
            r=s;
        }
        pb=pb-next;
    }
}
void c1_2(Linklist A, Linklist B, Linklist &C)
{
    LNode *pa=A->next, *pb, *s,*r;
    C=( LNode*)malloc(sizeof(LNode));
    C->next=null;
    r=C;
    while(pa)
    {   pb=B->next;
        while(pb&&pb->data!=pa->data)
            pb=pb->next;
```

```
        if(pb==null)
        {
            s=( LNode*)malloc(sizeof(LNode));
            s->data=pa->data;s->next=null;
            r->next=s;
            r=s;
        }
        pa=pa-next;
    }
}

int ListLength(LinkList L)
 { /* 初始条件：线性表L已存在。操作结果：返回L中数据元素个数 */
    int i=0;
    LinkList p=L->next; /* p指向第一个结点 */
    while(p) /* 没到表尾 */
    {
      i++;
      p=p->next;
    }
    return i;
 }
二、
Typedef struct Lnode{
    ElemType data;
    char sex;
    floar stature;
    struct LNode *next;
}LNode,*Linklist;
void DisCreat(LinkedList A)
{
    LinkedList B,C;
    LNode *p,*r,*b1,*b2,*c1,*c2;
    B=C=null;
    p=A; //p为工作指针。
    while(p)
    {r=p->next; //暂存p的后继。
    if (p->sex=='M')//男生放入B表。
        {if(B==null){p->next=null;B=p;}
        else{b1=null;b2=B;
```

```
            while(b2&&p->stature>b2->stature)
            {b1=b2;b2=b2->next;}
            p->next=b2; b1->next=p;}
        }
    else//女生放入C表
        {if(C==null){p->next=null;C=p;}
        else{c1=null;c2=C;
            while(c2&&p->stature>c2->stature)
            {c1=c2;c2=c2->next;}
            p->next=c2; c1->next=p;}
        }
    p=r;//p指向新的待处理结点。
}
```

三、

采用输出受限的双端队列

```
void Train_Rearrange(char *train)
//这里用字符串 train 表示火车，'P'表示硬座，'H'表示硬卧，'S'表示软卧。
{
    r=train;
    InitDQueue(Q);
    while(*r)
    {
        if(*r='P')
        {
            不入队列，直接输出；
        }
        else if(*r='S')
        {
            EnDQueue(Q, *r, 0);//从头端入队
        }
        else
        {
            EnDQueue(Q, *r, 1);//从尾端入队
        }
        r++;
    }//while
    while(!DQueueEmpty(Q))
    {
        DeDQueue(Q);//从头端出队
    }//while
```

```
}//Train_Rearrange
```

四、

(1)

```
void compress (int A[n][n], int B[ ], int n)
{//将三对角矩阵 A[0..n-1, 0..n-1]三条对角线上的元素逐行存放于数组 B 中
  K=0;
  for (i=0; i<n; i++)
       for (j=0; j<n; j++)
          if (A[i][j]!=0) B[k++]= A[i][j];
}
```

(2) 已知 k 求 i、j 时，则下标的对应关系是：

i = (k+1)/3

j = k−2i

```
void uncompress (int B[ ], int A[n][n], int n)
{//由数组 B[0..3n-3]确定三对角矩阵 A[0..n-1, 0..n-1]
for (i=0; i<n; i++)
       for (j=0; j<n; j++)
            A[i][j] =0;
   for(k=0; k<=3*n-3; k++)
       { i=(k+1)/3;
         j=k-2*i;
       A[i][j]=B[k];
       }
}
```

(3)

```
void TransMatrix(int B[ ], int C[ ],int n)
{
      C[0]=B[0];
      For(k=1;k<n;k++)
      {     C[k+1]=B[k];
            k++;
            C[k-1]=B[k];
            k++;
            C[k]=A[k];
      }
}
```

五、

用二叉树表示该大家族，可以用根结点表示父或母（祖先），根结点的左子女表示配偶，配偶的右子女表示子女。这种二叉树可以看成类似树的孩子兄弟链表表示法；

根结点是父或母，根无右子女，左子女表示配偶，配偶的右子女（右子女的右子女等）均可看成兄弟姐妹（即父母的所有子女），这些结点又可成为新的双亲。首先递归查找某家族成员结点，若查找成功，要么其左子女是配偶，配偶的右子女及右子女的右子女等均为父母的子女，要么其右子女及右子女的右子女等均为父母的子女。

```
BiTree Search(BiTree t,ElemType parent)//在二叉树上查找值为 parent 的结点
    {if(t==null) return (null);              //二叉树上无 parent 结点
     else if(t->data==parent) return(t);    //查找成功
     p=Search(t->lchild, parent); p=Search(t->rchild, parent); }
    }//结束 Search
void PrintSons(BiTree t,ElemType p)//在二叉树上查找结点值为 p 的所有的子女
     {p=Serach(t,p);                          //在二叉树 t 上查找父结点 p
      if(p!=null)                             //存在父结点
        {
        if(p->lchild!=null){q=p->lchild; q=q->rchild;}
        //先指向其配偶结点，再找到第一个子女
        else q=p->rchild;
         while(q!=null) {printf(q->data); q=q->rchild;} //输出所有子女
        }
      }//结束 PrintSons
```

六、

该题可用求每对顶点间最短路径的FLOYD算法求解。求出每一顶点到其它顶点的最短路径。在每个顶点到其它顶点的最短路径中，选出最长的一条。因为有n个顶点，所以有n条，在这n条最长路径中找出最短一条，它的出发点为所求。

```
void Signal(AdjMatrix w,int n)
{for (k=1;k<=n;k++) //求任意两顶点间的最短路径
for (i=1;i<=n;i++)
for (j=1;j<=n;j++)
if (w[i][k]+w[k][j]<w[i][j]) w[i][j]=w[i][k]+w[k][j];
m=MAXINT; //设定m为机器内最大整数。
for (i=1;i<=n;i++) //求最长路径中最短的一条。
{s=0;
  for (j=1;j<=n;j++) //求从某地区i（1<=i<=n）到其它地区的最长路径。
  if (w[i][j]>s) s=w[i][j];
  if (s<=m) {m=s; k=i;}//在最长路径中，取最短的一条。m记最长路径，k记出发顶点的下标。
  Printf(“供应站应建在%d地区\n”,i);
}//for
}
```

七、

```
void FindInDegree(ALGraph G,int indegree[])
```

```
 { /* 求顶点的入度*/
    int i;
    ArcNode *p;
    for(i=0;i<G.vexnum;i++)
       indegree[i]=0; /* 赋初值 */
    for(i=0;i<G.vexnum;i++)
    {
       p=G.vertices[i].firstarc;
       while(p)
       {
          indegree[p->adjvex]++;
          p=p->nextarc;
       }
    }
 }
 typedef int SElemType; /* 栈类型 */
Status TopologicalSort(ALGraph G)
 { /* 有向图 G 采用邻接表存储结构。若 G 无回路,则输出 G 的顶点的一
个拓扑序列并返回 OK,否则返回 ERROR。*/
    int i,k,count,indegree[MAX_VERTEX_NUM],term;
    SqStack S1,S2;
    ArcNode *p;
    FindInDegree(G,indegree); /* 对各顶点求入度 indegree[0..vernum-1] */
    InitStack(&S1); /* 初始化栈 */
     InitStack(&S2);
    for(i=0;i<G.vexnum;++i) /* 建零入度顶点栈 S */
       if(!indegree[i])
          Push(&S1,i); /* 入度为 0 者进栈 */
    count=0; /* 对输出顶点计数 */
     term=1;//学期
    while(!StackEmpty(S1))
    { /* 栈不空 */
       while(!StackEmpty(S1))
          {Pop(&S,&i); Push(&S2,i)}
      if (!StackEmpty(S2))
     { printf("第%d 个学期的课程有：",term);
      StackTraverse(S2,print);
        term++;
     }
     while(!StackEmpty(S2))
```

```
        {
        ++count;
        for(p=G.vertices[i].firstarc;p;p=p->nextarc)
        { /*  对 i 号顶点的每个邻接点的入度减 1 */
          k=p->adjvex;
          if(!(--indegree[k])) /*  若入度减为 0,则入栈  */
            Push(&S1,k);
        }
        }
    }
    if(count<G.vexnum) return ERROR;
    else    return OK;
 }
```

八、

该问题类似于五叉路口交通灯的设计，采用无向图的数据结构，将每个项目视为顶点，凡是一个运动员同时报名参加的项目，表示不能在同一单位时间进行，则用边连接发生冲突的顶点，然后进行染色分析。

```
//判断图 g 中的第 i 个结点是否与所有已着色为 count 的结点都不相邻
bool isnotadjacent(ALGraph& g, int i, int count)
{
      EdgeNode*  p=g.vertices[i].firstedge ;   //p 指向邻接表第 i 个结点的链表头
      while(p){
            if(g.vertices [p->adjvex].color ==count)
                  return false;
            p=p->nextedge;
      }
      return true;
}

//判断图 g 是否已全部着色
bool isallcolored(ALGraph& g)
{
      for(int i=0;i<g.vexnum;i++)
            if(g.vertices[i].color==0)
                  return false;
      return true;
}

//依图中结点的度数从大到小的顺序给图着色,返回着色数
```

```
int color(ALGraph& g)
{
    int i;
    int count=0;   //记录着色数
    while(!isallcolored(g))
    {
        count++;
        for(i=0;i<g.vexnum;i++)
            if(g.vertices[i].color==0){
                g.vertices[i].color=count;
                break;
            }   //先找到第一个未着色的结点进行着色
        for(i++;i<g.vexnum;i++){
            //如果第 i 个结点尚未着色并且该结点与刚着色为 count 的
结点
            //都不相邻,则将该结点着色为 count
            if(g.vertices[i].color==0 && isnotadjacent(g,i,count))
                g.vertices [i].color=count;
        }//for
    }//while
    return count;
}
```