第 1 题采用邻接矩阵构造无向网代码参考如下：

```
#include <stdio.h>
#include <stdlib.h>
#include <limits.h>
#define TRUE 1
#define FALSE 0
#define OK 1
#define ERROR 0
#define OVERFLOW -2
#define INFINITY INT_MAX          //最大值
#define MAX_VERTEX_NUM 20         //最大顶点个数
typedef int Status;
typedef int VRType;
typedef int InfoType;
typedef enum {DG, DN, UDG, UDN} GraphKind;     //{有向图，有向网，无向图，无向网}
typedef struct ArcCell {
      VRType adj;              //VRType 是顶点关系类型。对无权图，用 1 或 0
                               //表示相邻否；对带权图，则为权值类型。
      InfoType *info;       //该弧相关信息的指针
}ArcCell, AdjMatrix[MAX_VERTEX_NUM][MAX_VERTEX_NUM];
typedef char VertexType;
typedef struct {
      VertexType vexs[MAX_VERTEX_NUM];       //顶点向量
      AdjMatrix arcs;                                  //邻接矩阵
      int vexnum, arcnum;                              //图的当前顶点数和弧数
      GraphKind kind;                                  //图的种类标志
}MGraph;

int LocateVex(MGraph G, char v) {
      int i;
      for(i=0; i<G.vexnum; ++i)
            if (G.vexs[i]==v) return i;
      return -1;
}

Status CreateUDN(MGraph &G) {//    算法  7.2
   // 采用数组（邻接矩阵）表示法，构造无向网 G。
   int i,j,k,w;
   VertexType    v1,v2;
   printf("输入顶点数 G.vexnum: " );    scanf("%d", &G.vexnum);
   printf("输入边数 G.arcnum: ");      scanf("%d", &G.arcnum);
   getchar();   /*** 加上此句 getchar()!!! ***/
   // scanf("%d,%d,%d",&G.vexnum, &G.arcnum, &IncInfo);
   for (i=0; i<G.vexnum; i++) {
      printf("输入顶点 G.vexs[%d]: ",i);
      scanf("%c",&G.vexs[i]);
      getchar();
   } // 构造顶点向量
   for (i=0; i<G.vexnum; ++i)   // 初始化邻接矩阵
      for (j=0; j<G.vexnum; ++j) {
```

```
            G.arcs[i][j].adj = INFINITY; //{adj,info}
            G.arcs[i][j].info= NULL;
        }
    for (k=0; k<G.arcnum; ++k) {    // 构造邻接矩阵
        printf("输入第%d 条边 vi、vj 和权值 w (int): \n", k+1);
        scanf("%c %c %d", &v1, &v2, &w);     // 输入一条边依附的顶点及权值
        getchar();
        i = LocateVex(G, v1);    j = LocateVex(G, v2);       // 确定 v1 和 v2 在 G 中位置
        G.arcs[i][j].adj = w;                              // 弧<v1,v2>的权值
        // if (IncInfo) scanf(G.arcs[i][j].info); // 输入弧含有相关信息
        G.arcs[j][i].adj = G.arcs[i][j].adj; // 置<v1,v2>的对称弧<v2,v1>
    }
    return OK;
} // CreateUDN

Status CreateGraph(MGraph &G) {    // 算法 7.1
    // 采用数组(邻接矩阵)表示法，构造图 G。
    printf("请输入图的种类：0 表示 DG，1 表示 DN，2 表示 UDG，3 表示 UDN\n");
    scanf("%d", &G.kind);   // 自定义输入函数，读入一个随机值
    switch (G.kind) {
        case   DG: return CreateDG(G);     // 构造有向图 G
        case   DN: return CreateDN(G);     // 构造有向网 G
        case UDG: return CreateUDG(G);   // 构造无向图 G
        case UDN: return CreateUDN(G);   // 构造无向网 G，算法 7.2
        default : return ERROR;
    }
} // CreateGraph

void list(MGraph G) {
    int i, j;
    printf("输出邻接矩阵：\n");
    for(i=0; i<G.vexnum; ++i) {
        printf("%c----", G.vexs[i]);
        for(j=0; j<G.vexnum; ++j)
            if (G.arcs[i][j].adj==INFINITY)
                printf("%4s", "∞");
            else
                printf("%4d", G.arcs[i][j].adj);
        printf("\n");
    }
}

void main() {
    MGraph G;
    CreateGraph(G);
    list(G);
}
```

第 2 题采用邻接表构造无向图代码参考如下：

```
#include <stdio.h>
#include <stdlib.h>
#define TRUE 1
#define FALSE 0
#define OK 1
#define ERROR 0
#define OVERFLOW -2
#define MAX_VERTEX_NUM 20        //最大顶点个数
typedef int Status;
typedef int InfoType;
typedef struct ArcNode {
    int adjvex;                  //该弧所指向的顶点位置
    struct ArcNode *nextarc;     //指向下一条弧的指针
    InfoType *info;              //该弧相关信息的指针
}ArcNode;
typedef char VertexType;
typedef struct VNode {
    VertexType data;             //顶点信息
    ArcNode *firstarc;           //指向第一条依附该顶点的弧的指针
}VNode, AdjList[MAX_VERTEX_NUM];
typedef struct {
    AdjList vertices;
    int vexnum, arcnum;          //图的当前顶点数和弧数
    int kind;                    //图的种类标志
}ALGraph;

int LocateVex(ALGraph G, char v) {
    int i;
    for(i=0; i<G.vexnum; ++i)
        if (G.vertices[i].data==v) return i;
    return -1;
}

Status CreateUDG(ALGraph &G) {
    // 采用邻接表存储表示，构造无向图 G（G.kind=UDG）。
    int i, j, k, IncInfo;
    ArcNode *pi, *pj;
    char v1, v2;
    //scanf("%d %d &d", &G.vexnum, &G.arcnum, &IncInfo);        // IncInfo 为 0 表明各弧不含其它信息
    printf("输入顶点数 G.vexnum: " );   scanf("%d",&G.vexnum);
    printf("输入边数 G.arcnum: ");     scanf("%d",&G.arcnum);
    printf("输入边包含其它信息情况(1--包含，0--不包含)： ");     scanf("%d",&IncInfo);
    getchar();
    for (i=0; i<G.vexnum; ++i) {                        // 构造表头向量
        printf("输入顶点 G.vertices[%d].data: ",i);
        scanf("%c", &G.vertices[i].data);               // 输入顶点值
        getchar();
        G.vertices[i].firstarc = NULL;                  // 初始化链表头指针为"空"
    }//for
```

```
    for (k=0; k<G.arcnum; ++k) {                    // 输入各边并构造邻接表
        printf("请输入第%d 条边的两个顶点：\n", k+1);
        scanf("%c%c",&v1, &v2);                     // 输入一条边的始点和终点
        getchar();
        i = LocateVex(G, v1);   j = LocateVex(G, v2);    // 确定 v1 和 v2 在 G 中位置，即顶点的序号
        if (!(pi = (ArcNode *)malloc(sizeof(ArcNode)))) exit(OVERFLOW);
        pi -> adjvex = j;                           // 对弧结点赋邻接点"位置"信息
        pi -> nextarc = G.vertices[i].firstarc;   G.vertices[i].firstarc = pi;
                                                    // 插入链表 G.vertices[i]
        if (!(pj = (ArcNode *)malloc(sizeof(ArcNode)))) exit(OVERFLOW);
        pj -> adjvex = i;                           // 对弧结点赋邻接点"位置"信息
        pj -> nextarc = G.vertices[j].firstarc;   G.vertices[j].firstarc = pj;
                                                    // 插入链表 G.vertices[j]
        /* if (IncInfo) {                           // 若弧含有相关信息，则输入
            printf("请输入弧包含的相关信息情况：\n");
            scanf("%d", pj->info);
            pi->info=pj->info;
        }  */
    }//for
    return OK;
} // CreateUDG

Status CreateGraph(ALGraph &G) {   //
    // 采用邻接表，构造图 G。
    printf("请输入图的种类：0 表示 DG，1 表示 DN，2 表示 UDG，3 表示 UDN \n");
    scanf("%d", &G.kind);   // 自定义输入函数，读入一个随机值
    switch (G.kind) {
        case 0: return CreateDG(G);     // 构造有向图 G
        case 1: return CreateDN(G);     // 构造有向网 G
        case 2: return CreateUDG(G);    // 构造无向图 G
        case 3: return CreateUDN(G);    // 构造无向网 G
        default : return ERROR;
    }
} // CreateGraph

void list(ALGraph G) {
    int i;
    ArcNode *p;
    printf("输出邻接表：\n");
    for(i=0; i<G.vexnum; ++i) {
        printf("%d：%c--->", i, G.vertices[i].data);
        p = G.vertices[i].firstarc;
        while(p) {
            printf("%3d", p->adjvex);
            p = p->nextarc;
        }
        printf("\n");
    }
}
```

```
void main() {
	ALGraph G;
	CreateGraph(G);
	list(G);
}
```