Assignment 9

Answer 1:

```
#include <stdio.h>
#include <stdlib.h>

struct Node{
   int v;
   struct Node* next;
};

struct Node* adj[100];
int vis[100];

void addEdge(int u,int v){
   struct Node* p=malloc(sizeof(struct Node));
   p->v=v; p->next=adj[u]; adj[u]=p;
}

void bfs(int start,int n){
   int q[100],f=0,r=0;
   for(int i=0;i<n;i++) vis[i]=0;
   vis[start]=1;
   q[r++]=start;
   while(f<r){
      int u=q[f++];
      printf("%d ",u);
      struct Node* t=adj[u];
      while(t){
         if(!vis[t->v]){
            vis[t->v]=1;
            q[r++]=t->v;
         }
         t=t->next;
      }
   }
}
```

Answer 2:

```
void dfsUtil(int u){
   vis[u]=1;
```

```
    printf("%d ",u);
    struct Node* t=adj[u];
    while(t){
        if(!vis[t->v]) dfsUtil(t->v);
        t=t->next;
    }
}

void dfs(int start,int n){
    for(int i=0;i<n;i++) vis[i]=0;
    dfsUtil(start);
}
```

Answer 3 (Kruskal (Using Union–Find)):

```
struct Edge{
    int u,v,w;
};

int parent[100];

int find(int x){
    if(parent[x]==x) return x;
    return parent[x]=find(parent[x]);
}

void uni(int x,int y){
    x=find(x);
    y=find(y);
    parent[y]=x;
}

void kruskal(struct Edge e[],int m,int n){
    for(int i=0;i<n;i++) parent[i]=i;

    for(int i=0;i<m-1;i++){
        for(int j=i+1;j<m;j++){
            if(e[j].w < e[i].w){
                struct Edge t=e[i]; e[i]=e[j]; e[j]=t;
            }
        }
    }

    for(int i=0;i<m;i++){
```

```
        int a=find(e[i].u);
        int b=find(e[i].v);
        if(a!=b){
            printf("%d-%d(%d) ",e[i].u,e[i].v,e[i].w);
            uni(a,b);
        }
    }
}
```

Answer 3 (Prim's Algorithm (Adjacency Matrix)):

```
int prim(int g[100][100],int n){
    int key[100],used[100],i,j;
    for(i=0;i<n;i++){ key[i]=999999; used[i]=0; }
    key[0]=0;
    int res=0;

    for(i=0;i<n;i++){
        int u=-1;
        for(j=0;j<n;j++){
            if(!used[j] && (u==-1 || key[j]<key[u])) u=j;
        }
        used[u]=1;
        res+=key[u];
        for(j=0;j<n;j++){
            if(g[u][j] && !used[j] && g[u][j] < key[j]){
                key[j]=g[u][j];
            }
        }
    }
    return res;
}
```

Answer 4:

```
int dijkstra(int g[100][100],int n,int src,int dist[]){
    int vis[100];
    for(int i=0;i<n;i++){ dist[i]=999999; vis[i]=0; }
    dist[src]=0;

    for(int k=0;k<n;k++){
        int u=-1;
        for(int i=0;i<n;i++){
            if(!vis[i] && (u==-1 || dist[i]<dist[u])) u=i;
```

```
        }
        vis[u]=1;
        for(int v=0;v<n;v++){
            if(g[u][v] && dist[u]+g[u][v] < dist[v]){
                dist[v]=dist[u]+g[u][v];
            }
        }
    }
    return 0;
}
```