```
/*
Assignment 1:
Represent a given graph G=(V,E) in a computer that can be of the following category:
i)  Undirected and Unweighted
ii) Undirected and Weighted
iii)Directed and Unweighted
iv) Directed and Weighted
Using all possible techniques of representing a graph in a computer as:
i)  Adjacency Matrix
ii) Incidence Matrix
iii)Adjacency List
*/

/*Include the header files*/
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>

/*Forward declaration of the adjacent vertex strcture*/
struct sub_node;

/*Structure of the main vertex*/
struct main_node
{
    int ver;
    struct main_node *nextver;
    struct sub_node *adver;
};

/*Structure of the adjacent vertex*/
struct sub_node
{
    int wt;
    struct main_node *vert;
    struct sub_node *next;
};

/*Fuction to create a memory allocation for the main vertex*/
struct main_node* getmain(int x)
{
    struct main_node *new1;
    new1=(struct main_node *)malloc(sizeof(struct main_node));
    new1->ver=x;
    new1->nextver=NULL;
    new1->adver=NULL;
    return(new1);
}

/*Function to create a memory allocation for the adjacent vertex*/
struct sub_node* getsub(int x)
{
    struct sub_node *new1;
    new1=(struct sub_node *)malloc(sizeof(struct sub_node));
    new1->wt=x;
    new1->vert=NULL;
    new1->next=NULL;
    return(new1);
}

/*Declarations of the global variables for Adjacent Matrix,Incidence Matrix,Adjacent List,Numb
er of Vertex and Number of Edge*/
struct main_node *head;
int admt[50][50],inmt[50][50],n,m;

int main()
{
    /*Declaration od the prototypes of the functions to be used*/
    void adjacency_matrix_create(int,int);
    void incidence_matrix_create(int,int);
    void adjacency_list_create(int,int);
    void display(int,int);
    int c;
    /*Loop for user's choice for the type of graph user wants to enter*/
    do
    {
        printf("\tType of Graph");
        printf("\n1.UnDirect & UnWeighted");
        printf("\n2.UnDirect & Weighted");
        printf("\n3.Direct & UnWeighted");
        printf("\n4.Direct & Weighted");
        printf("\n5.Exit Program ");
        printf("\nEnter choice (1,2,3,4,5):- ");
```

```
        scanf("%d",&c);
        switch(c)
        {
            /*Case for UnDirect & UnWeighted Graph*/
            case 1:
                adjacency_matrix_create(0,0);
                incidence_matrix_create(0,0);
                adjacency_list_create(0,0);
                display(0,0);
                break;
            /*Case for UnDirect & Weighted Graph*/
            case 2:
                adjacency_matrix_create(0,1);
                incidence_matrix_create(0,1);
                adjacency_list_create(0,1);
                display(0,1);
                break;
            /*Case for Direct & UnWeighted Graph*/
            case 3:
                adjacency_matrix_create(1,0);
                incidence_matrix_create(1,0);
                adjacency_list_create(1,0);
                display(1,0);
                break;
            /*Case for Direct & Weighted Graph*/
            case 4:
                adjacency_matrix_create(1,1);
                incidence_matrix_create(1,1);
                adjacency_list_create(1,1);
                display(1,1);
                break;
            case 5:
                exit(0);
            default:
                printf("\nWrong Choice : Enter again\n");
                continue;
        }
    }while(1);
    return 0;
}

/*Function to create the adjacency matrix when user is givin the input*/
void adjacency_matrix_create(int x,int y)
{
    int i,j,c;
    printf("Enter number of vertices :- ");
    scanf("%d",&n);
    for(i=0;i<n;i++)
    {
        /*If undirected*/
        if(x == 0)
            j=i;
        /*If directed*/
        else
            j=0;
        for(;j<n;j++)
        {
            if(i==j)
                admt[i][j]=0;
            else
            {
                do
                {
                    /*If unweighted*/
                    if(y == 0)
                    {
                        printf("Does edge (%d , %d) exist ?(1=YES,0=NO) :- ",i+1,j+1);
                        scanf("%d",&c);
                        if(c != 0 && c != 1)
                        {
                            printf("Wrong Input : Enter 0=NO or 1=YES\n");
                            continue;
                        }
                        else
                            break;
                    }
                    /*If weighted*/
                    else
                    {
                        printf("Does edge (%d , %d) exist ?If YES enter weight else (0=NO) :-
",i+1,j+1);
                        scanf("%d",&c);
```

```
                        if(c < 0)
                        {
                            printf("Wrong Input : Enter 0=NO or >0 for weight\n");
                            continue;
                        }
                        else
                            break;
                    }
                }while(1);
                admt[i][j]=c;
                /*If undirected*/
                if(x == 0)
                    admt[j][i]=c;
            }
        }
    }
}

/*Function to create the incidence matrix from the adjacent matrix*/
void incidence_matrix_create(int x,int y)
{
    int i,j,z;
    z=n*(n-1)/2;
    m=0;
    for(i=0;i<n;i++)
        for(j=0;j<z;j++)
            inmt[i][j]=0;
    for(i=0;i<n;i++)
    {
        /*If undirected*/
        if(x == 0)
            j=i+1;
        /*If directed*/
        else
            j=0;
        for(;j<n;j++)
        {
            /*If unweighted*/
            if(y == 0)
            {
                if(admt[i][j] == 1)
                {
                    inmt[i][m]=1;
                    /*If undirected*/
                    if(x == 0)
                        inmt[j][m]=1;
                    /*If directed*/
                    else
                        inmt[j][m]=-1;
                    m++;
                }
            }
            /*If weighted*/
            else
            {
                if(admt[i][j] > 0)
                {
                    inmt[i][m]=admt[i][j];
                    /*If undirected*/
                    if(x == 0)
                        inmt[j][m]=admt[i][j];
                    /*If directed*/
                    else
                        inmt[j][m]=-1;
                    m++;
                }
            }
        }
    }
}

/*Function that creates the adjacency list from tha adjacency matrix*/
void adjacency_list_create(int x,int y)
{
    struct main_node *new1,*ptr1,*ptr2;
    struct sub_node *new2,*ptrr1;
    int i=0,j,c,f;
    head=NULL;
    do{
        /*Creating the vertex list*/
        new1=getmain(i++);
        if(head == NULL)
```

```
            head=new1;
        else
        {
            ptr1=head;
            while(ptr1->nextver != NULL)
                ptr1=ptr1->nextver;
            ptr1->nextver=new1;
        }
    }while(i < n);
    ptr1=head;
    while(ptr1 != NULL)
    {
        i=ptr1->ver;
        /*If undirected*/
        if(x == 0)
            j=i+1;
        /*If directed*/
        else
            j=0;
        for(;j<n;j++)
        {
            /*If unweighted*/
            if(y == 0)
            {
                if(admt[i][j] == 1)
                {
                    f=1;
                    c=0;
                }
                else
                    f=0;
            }
            /*If weighted*/
            else
            {
                if(admt[i][j] > 0)
                {
                    f=1;
                    c=admt[i][j];
                }
                else
                    f=0;
            }
            if(f == 1)
            {
                new2=getsub(c);
                ptr2=head;
                while(ptr2->ver != j)
                    ptr2=ptr2->nextver;
                new2->vert=ptr2;
                if(ptr1->adver == NULL)
                    ptr1->adver=new2;
                else
                {
                    ptrr1=ptr1->adver;
                    while(ptrr1->next != NULL)
                        ptr1=ptrr1->next;
                    ptrr1->next=new2;
                }
                /*If undirected*/
                if(x == 0)
                {
                    new2=getsub(c);
                    new2->vert=ptr1;
                    if(ptr2->adver == NULL)
                        ptr2->adver=new2;
                    else
                    {
                        ptrr1=ptr2->adver;
                        while(ptrr1->next != NULL)
                            ptrr1=ptrr1->next;
                        ptrr1->next=new2;
                    }
                }
            }
        }
        ptr1=ptr1->nextver;
    }
}

/*Function to display the graph entered on user's choice*/
void display(int x,int y)
```

```
{
    struct main_node *ptr1;
    struct sub_node *ptrr1;
    int i,j,c;
    do
    {
        printf("\n\tGraph Representation");
        printf("\n1.Adjacency Matrix");
        printf("\n2.Incidence Matrix");
        printf("\n3.Adjacency List");
        printf("\n4.Return to Main");
        printf("\n5.Exit Program");
        printf("\nEnter choice (1,2,3,4,5):- ");
        scanf("%d",&c);
        switch(c)
        {
            /*Adjacency Matrix Representation*/
            case 1:
                printf("\nAdjacency Matrix\n");
                printf("  ");
                for(i=0;i<n;i++)
                    printf("%d ",i+1);
                printf("\n");
                for(i=0;i<n;i++)
                {
                    printf("%d\t",i);
                    for(j=0;j<n;j++)
                        printf("%d\t",admt[i][j]);
                    printf("\n");
                }
                break;
            /*Incidence Matrix Representation*/
            case 2:
                printf("\nIncidence Matrix\n");
                printf("  ");
                for(i=0;i<m;i++)
                    printf("%d\t",i+1);
                printf("\n");
                for(i=0;i<n;i++)
                {
                    printf("%d\t",i+1);
                    for(j=0;j<m;j++)
                        printf("%d\t",inmt[i][j]);
                    printf("\n");
                }
                break;
            /*Adjacency List Representation*/
            case 3:
                printf("\nAdjacency List\n");
                ptr1=head;
                if(y == 0)
                    printf("\nVertex:\tAdjacent Vertices\n");
                else
                    printf("\nVertex\tAdjacent Vertices and weights\n");
                while(ptr1 != NULL)
                {
                    printf("%d\t:",(ptr1->ver)+1);
                    ptrr1=ptr1->adver;
                    while(ptrr1 != NULL)
                    {
                        if(y == 0)
                            printf("%d,\t",(ptrr1->vert->ver)+1);
                        else
                            printf("%d(%d),\t",(ptrr1->vert->ver)+1,(ptrr1->wt));
                        ptrr1=ptrr1->next;
                    }
                    printf("\n");
                    ptr1=ptr1->nextver;
                }
                break;
            case 4:
                /*Return to main*/
                return;
            case 5:
                exit(0);
            default:
                    printf("\nWrong Choice : Enter again\n");
                    continue;
        }
    }while(1);
}
```