```
/*
Assignment 3:
a)	Perform BFS traversal on a Graph G=(V,E) where
i)	G is undirected
ii)	G is directed
and compute its BFS tree.
b)	Compute the shortest distance between a pair of  vertices of a given Graph
G=(V,E) where
i)		G is undirected
i)	G is directed
*/

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

/*Forward declaration of the adjacent vertex structure*/
struct subvert;

/*Structure of the main vertex*/
struct mainvert
{
	int ver;
	int dst;
	int visited;
	struct mainvert *nextver;
	struct subvert *adver;
};

/*Structure of the adjacent vertex*/
struct subvert
{
	struct mainvert *vert;
	struct subvert *next;
};

/*Fuction to create a memory allocation for the main vertex*/
struct mainvert *getmain(int x)
{
	struct mainvert *new1;
	new1=(struct mainvert *)malloc(sizeof(struct mainvert));
	new1->ver=x;
	new1->dst=0;
	new1->visited=0;
	new1->nextver=NULL;
```

```
	new1->adver=NULL;
	return(new1);
}

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

/*Declarations of the global variables for Adjacency List and Number of Components*/
struct mainvert *head;
int n,m;

int main()
{
	/*Declaration od the prototypes of the functions to be used*/
	void adjacency_list_create(int);
	void bfs_trav(int,int,int);
	void display();
	void shrstdst(int);
	int s,c1,c2,x=0,y;
	/*Loop for user's choice to perform BFS traversal or find shortest distance
between 2 vertices*/
	do
	{
		/*Main menu for the operation to be performed*/
		printf("\tMAIN MENU");
		printf("\n1.BFS Traversal");
		printf("\n2.Shortest Distance between 2 vertices");
		printf("\n3.EXIT");
		printf("\nEnter your choice (1,2.3):- ");
		scanf("%d",&c1);
		switch(c1)
		{
			/*Sub menu for BFS of undirected or directed graph*/
			case 1:
				printf("\tSUB MENU");
				printf("\n1.UnDirected");
				printf("\n2.Directed");
				printf("\nEnter your choice (1,2):- ");
				scanf("%d",&c2);
```

```
		switch(c2)
		{
			/*UnDirected Graph*/
			case 1:
				y=0;
				adjacency_list_create(0);
				display();
				printf("\nEnter starting node :- ");
				scanf("%d",&s);
				bfs_trav(y,s,0);
				break;
			/*Directed Graph*/
			case 2:
				y=1;
				adjacency_list_create(1);
				display();
				printf("\nEnter starting node :- ");
				scanf("%d",&s);
				bfs_trav(y,s,0);
				break;
			default:
				printf("\nWrong Input");
				break;
		}
		break;
/*Sub menu for shortest distance of undirected or directed graph*/
case 2:
		printf("\tSUB MENU");
		printf("\n1.UnDirected");
		printf("\n2.Directed");
		printf("\nEnter your choice (1,2):- ");
		scanf("%d",&c2);
		switch(c2)
		{
			/*UnDirected Graph*/
			case 1:
				if(x == 0)
				{
					y=0;
					adjacency_list_create(y);
					display();
				}
				if(y == 0)
					shrstdst(0);
				else
```

```
                                                printf("\nEntered Graph is of
Directed type .....\n");
                                        break;
                                /*Directed Graph*/
                                case 2:
                                        if(x == 0)
                                        {
                                                y=1;
                                                adjacency_list_create(y);
                                                display();
                                        }
                                        if(y == 1)
                                                shrstdst(1);
                                        else
                                                printf("\nEntered Graph is of
Undirected type .....\n");
                                        break;
                                default:
                                        printf("\nWrong Input");
                                        break;

                        }
                        break;
                case 3:
                        exit(0);
                default:
                        printf("\nWrong Input");
                        break;
                }
                x++;
        }while(1);
        return 0;
}

/*Function that creates the adjacency list entered by the user*/
void adjacency_list_create(int x)
{
        struct mainvert *new1,*ptr,*ptr1;
        struct subvert *new2,*ptrr1;
        int f1,c,a;
        n=0;
        head=NULL;
        do
        {
                /*Creating the vertex list*/
                n++;
```

```
                new1=getmain(n);
                if(head == NULL)
                        head=new1;
                else
                {
                        ptr=head;
                        while(ptr->nextver != NULL)
                                ptr=ptr->nextver;
                        ptr->nextver=new1;
                }
                printf("Vertex %d created",n);
                printf("\nDo you want to add any more vertex?(YES=1,NO=0) :- ");
                scanf("%d",&c);
        }while(c == 1);
        /*Entering the adjacent vertices*/
        printf("Enter the adjacent vertices of the vertices\n");
        ptr=head;
        while(ptr != NULL)
        {
                f1=1;
                c=0;
                /*Checking if there are any vertices of the vertex whose adjacent vertices
are to be enterde*/
                if(ptr->adver != NULL)
                {
                        printf("Vertices adjacent to %d are : ",ptr->ver);
                        ptrr1=ptr->adver;
                        while(ptrr1 != NULL)
                        {
                                printf("%d\t",ptrr1->vert->ver);
                                ptrr1=ptrr1->next;
                        }
                        printf("\nDoes %d have any more adjacent
vertices?(YES=1,NO=0) :- ",ptr->ver);
                        scanf("%d",&c);
                        if(c == 0 || c > 1)
                                f1=0;
                }
                /*If the vertex has adjacent vertices then those are entered by the user*/
                if(f1 == 1)
                {
                        do
                        {
                                if(c == 1)
                                {
```

```
                                        printf("Enter the vertex adjacent to %d (else 0):-
",ptr->ver);
                                        scanf("%d",&a);
                                        c=0;
                                }
                                else
                                {
                                        printf("Enter the vertex adjacent to %d (if no
adjacent vertex enter 0):- ",ptr->ver);
                                        scanf("%d",&a);
                                }
                                if(a == 0)
                                        break;
                                ptr1=head;
                                while(ptr1 != NULL && ptr1->ver != a)
                                        ptr1=ptr1->nextver;
                                if(ptr1 == NULL)
                                {
                                        printf("\nWrong Input : Re-Enter\n");
                                        continue;
                                }
                                ptrr1=ptr->adver;
                                while(ptrr1 != NULL && ptrr1->vert->ver != a)
                                        ptrr1=ptrr1->next;
                                if(ptrr1 != NULL)
                                {
                                        if(ptrr1->vert->ver == a)
                                                printf("ERROR : %d is already adjacent to
%d\n",a,ptr->ver);
                                }
                                else
                                {
                                        new2=getsub();
                                        new2->vert=ptr1;
                                        if(ptr->adver == NULL)
                                                ptr->adver=new2;
                                        else
                                        {
                                                ptrr1=ptr->adver;
                                                while(ptrr1->next != NULL)
                                                        ptrr1=ptrr1->next;
                                                ptrr1->next=new2;
                                        }
                                        if(x == 0)
                                        {
                                                new2=getsub();
```

```
                                        new2->vert=ptr;
                                        if(ptr1->adver == NULL)
                                                ptr1->adver=new2;
                                        else
                                        {
                                                ptrr1=ptr1->adver;
                                                while(ptrr1->next != NULL)
                                                        ptrr1=ptrr1->next;
                                                ptrr1->next=new2;
                                        }
                                }
                        }
                }while(1);
        }
        ptr=ptr->nextver;
    }
}

/*Funtion for the BFS traversal of the entered graph*/
void bfs_trav(int x,int s,int m)
{
    struct mainvert *queue[50],*ptr,*ptr1,*temp;
    struct subvert *ptrr1;
    int front,rear,c,min,d;
    /*Entering the starting vertex*/
    if(m == 0)
        printf("\nBFS Traversal:-");
    do
    {
        ptr=head;
        while(ptr->ver != s)
            ptr=ptr->nextver;
        if(ptr == NULL)
        {
            printf("Wrong Input : %d is not a vertex of the given graph\n->To
re-enter press 1 else 0 :- ",s);
            scanf("%d",&c);
            if(c == 0)
                break;
        }
        else
            break;
        printf("\nEnter starting node :- ");
        scanf("%d",&s);
    }while(1);
    printf("\n");
```

```
        /*Initializing variables*/
        d=0;
        front=-1;
        rear=-1;
        queue[++front]=ptr;
        queue[++rear]=ptr;
        ptr->dst=d++;
        ptr->visited=1;
        do
        {       /*Performing the BFS operation and giving the distance from the source
vertex*/
                temp=queue[front];
                if(m == 0)
                        printf("%d\t",temp->ver);
                ptrr1=temp->adver;
                while(ptrr1 != NULL)
                {
                        if(ptrr1->vert->visited == 0)
                        {
                                queue[++rear]=ptrr1->vert;
                                ptrr1->vert->visited=1;
                                ptrr1->vert->dst=d;
                        }
                        ptrr1=ptrr1->next;
                }
                d++;
                front++;
                /*Break infinite loop condition and for directed graph to find the next
minimum vertex of the vertex traversed if it is a dead end*/
                if(front > rear)
                {
                        if(x == 0)
                                break;
                        else
                        {
                                ptr=head;
                                min=ptr->ver;
                                while(ptr != NULL)
                                {
                                        if(min > ptr->ver && ptr->visited == 0)
                                        {
                                                ptr1=ptr;
                                                min=ptr->ver;
                                        }
                                        ptr=ptr->nextver;
                                }
```

```
                                        if(ptr != NULL)
                                        {
                                                queue[front]=ptr1;
                                                rear++;
                                        }
                                        else
                                                break;
                                }
                        }
                }while(1);
                printf("\n");
}

/*Function to find the shortest ditance between 2 vertices*/
void shrstdst(int x)
{
        struct mainvert *ptr;
        int s,e,m;
        printf("\nFinding shortest distance between 2 vertices");
        printf("\nEnter starting node :- ");
        scanf("%d",&s);
        printf("Enter ending node :- ");
        scanf("%d",&e);
        ptr=head;
        while(ptr != NULL)
        {
                ptr->visited=0;
                ptr=ptr->nextver;
        }
        /*For the undirected graph*/
        if(x == 0)
        {
                bfs_trav(0,s,0);
                ptr=head;
                while(ptr != NULL && ptr->ver != e)
                        ptr=ptr->nextver;
                if(ptr == NULL)
                        printf("\nWrong Input : %d is not a vertex of the given graph\n",e);
                else
                        printf("\nDistance between vertices (%d , %d) = %d\n",s,e,ptr-
>dst);
                return;
        }
        /*For the directed graph*/
        else
        {
```

```
                bfs_trav(1,s,0);
                ptr=head;
                while(ptr != NULL && ptr->ver != e)
                        ptr=ptr->nextver;
                if(ptr == NULL)
                {
                        printf("\nWrong Input : %d is not a vertex of the given graph\n",e);
                        return;
                }
                else
                        m=ptr->dst;
                ptr=head;
                while(ptr != NULL)
                {
                        ptr->visited=0;
                        ptr=ptr->nextver;
                }
                bfs_trav(1,e,1);
                ptr=head;
                while(ptr != NULL && ptr->ver != s)
                        ptr=ptr->nextver;
                if(ptr->dst == 0)
                        printf("Distance between vertices (%d , %d) = %d\n",s,e,m);
                else if(m < ptr->dst)
                        printf("Distance between vertices (%d , %d) = %d\n",s,e,m);
                else
                        printf("Distance between vertices (%d , %d) = %d\n",e,s,ptr->dst);
        }
}


/*Displaying the adjacency list*/
void display()
{
        struct mainvert *ptr1;
        struct subvert *ptrr1;
        /*Adjacency List Representation*/
        printf("\nAdjacency List\n");
        ptr1=head;
        printf("\nVertex:\tAdjacent Vertices\n");
        while(ptr1 != NULL)
        {
                printf("%d\t:",ptr1->ver);
                ptrr1=ptr1->adver;
                while(ptrr1 != NULL)
                {
```

```
                printf("%d,\t",ptrr1->vert->ver);
                ptrr1=ptrr1->next;
            }
            printf("\n");
            ptr1=ptr1->nextver;
        }
}
```