

```

//Implement Breadth First Search of a Graph
#include<stdio.h>
#include<stdlib.h>

#define MAX 100

#define initial 1
#define waiting 2
#define visited 3

int n;
int adj[MAX][MAX];
int state[MAX];
void create_graph();
void BF_Traversal();
void BFS(int v);

int queue[MAX], front = -1, rear = -1;
void insert_queue(int vertex);
int delete_queue();
int isEmpty_queue();

int main()
{
    create_graph();
    BF_Traversal();
    return 0;
}

void BF_Traversal()
{
    int v;

    for(v=0; v<n; v++)
        state[v] = initial;

    printf("Enter Start Vertex for BFS: \n");
    scanf("%d", &v);
    BFS(v);
}

void BFS(int v)
{
    int i;

    insert_queue(v);
    state[v] = waiting;

    while(!isEmpty_queue())
    {
        v = delete_queue();
        printf("%d ", v);
        state[v] = visited;

        for(i=0; i<n; i++)
        {
            if(adj[v][i] == 1 && state[i] == initial)
            {
                insert_queue(i);
                state[i] = waiting;
            }
        }
        printf("\n");
    }
}

void insert_queue(int vertex)
{
    if(rear == MAX-1)

```

```

        printf("Queue Overflow\n");
    else
    {
        if(front == -1)
            front = 0;
        rear = rear+1;
        queue[rear] = vertex ;
    }
}

int isEmpty_queue()
{
    if(front == -1 || front > rear)
        return 1;
    else
        return 0;
}

int delete_queue()
{
    int delete_item;
    if(front == -1 || front > rear)
    {
        printf("Queue Underflow\n");
        exit(1);
    }

    delete_item = queue[front];
    front = front+1;
    return delete_item;
}

void create_graph()
{
    int count,max_edge,origin,destin;

    printf("Enter number of vertices : ");
    scanf("%d",&n);
    max_edge = n*(n-1);

    for(count=1; count<=max_edge; count++)
    {
        printf("Enter edge %d( -1 -1 to quit ) : ",count);
        scanf("%d %d",&origin,&destin);

        if((origin == -1) && (destin == -1))
            break;

        if(origin>=n || destin>=n || origin<0 || destin<0)
        {
            printf("Invalid edge!\n");
            count--;
        }
        else
        {
            adj[origin][destin] = 1;
        }
    }
}

/*OUTPUT
Enter number of vertices : 9
Enter edge 1( -1 -1 to quit ) : 0 1
Enter edge 2( -1 -1 to quit ) : 0 3
Enter edge 3( -1 -1 to quit ) : 0 4
Enter edge 4( -1 -1 to quit ) : 1 2
Enter edge 5( -1 -1 to quit ) : 3 6
Enter edge 6( -1 -1 to quit ) : 4 7
Enter edge 7( -1 -1 to quit ) : 6 4
Enter edge 8( -1 -1 to quit ) : 6 7

```

```
Enter edge 9( -1 -1 to quit ) : 2 5
Enter edge 10( -1 -1 to quit ) : 4 5
Enter edge 11( -1 -1 to quit ) : 7 5
Enter edge 12( -1 -1 to quit ) : 7 8
Enter edge 13( -1 -1 to quit ) : -1 -1
Enter Start Vertex for BFS:
0
0 1 3 4 2 6 5 7 8
*/
```