```c
#include<stdio.h>
#include<stdlib.h>
#define size 40

struct node{
    int dest;
    struct node* next;
};
struct adj_list{
    struct node* head;

};
struct graph{
    int v;
    struct adj_list* array;
    int* visited;
    int* parent;
};

struct queue{
    int items[size];
    int front,rear;
};

struct node* cnode(int dest){
    struct node* temp=(struct node*)malloc(sizeof(struct node));
    temp->dest=dest;
    temp->next=NULL;
    return(temp);
}

struct graph* cgraph(int v){
    struct graph* gr=(struct graph*)malloc(sizeof(struct graph));
    gr->v=v;
    gr->array=(struct adj_list*)malloc(v*sizeof(struct adj_list));
    gr->visited=(int *)malloc(v*sizeof(int));
    gr->parent=(int*)malloc(v*sizeof(int));
    int i=0;
    for(i=0;i<v;++i){
        gr->array[i].head=NULL;
        gr->visited[i]=0;
        gr->parent[i]=-1;
    }
    return(gr);
}

void add_edge(struct graph* gr,int src,int dest){
    struct node* temp1=cnode(dest-1);
    temp1->next=gr->array[src-1].head;
    gr->array[src-1].head=temp1;
    struct node* temp2=cnode(src-1);
    temp2->next=gr->array[dest-1].head;
    gr->array[dest-1].head=temp2;
}
```

```

struct queue* cqueue(){
    struct queue* q=(struct queue*)malloc(sizeof(queue));
    q->front=-1;
    q->rear=-1;
    return(q);

}

int isEmpty(struct queue* q) {
    if(q->rear == -1)
        return 1;
     return 0;
}

void enqueue(struct queue* q, int value){
    if(q->rear == size-1)
        printf("\nQueue is Full!!");
    else {
        if(q->front == -1)
            q->front = 0;
        q->rear++;
        q->items[q->rear] = value;
    }
}

int dequeue(struct queue* q){
    int item;
    if(isEmpty(q)){
        printf("Queue is empty");
        item = -1;
    }
    else{
        item = q->items[q->front];
        q->front++;
        if(q->front > q->rear){
            q->front = q->rear = -1;
        }
    }
    return item;
}

void bfs(struct graph* gr,int start){
    struct queue*q=cqueue();
    gr->visited[start]=1;
    enqueue(q,start);
    while(!isEmpty(q)){
        int currentVertex=dequeue(q);
        struct node* temp=gr->array[currentVertex].head;
        while(temp){
            int t=temp->dest;
            if(gr->visited[t]==0){
                gr->parent[t]=currentVertex+1;
                gr->visited[t]=1;
```

```c
                enqueue(q,t);
            }
            temp=temp->next;
        }
    }
}

int main()
{   char ch;
    struct graph* gr = cgraph(10);
    add_edge(gr, 1,2);
    add_edge(gr, 1,3);
    add_edge(gr, 1,4);
    add_edge(gr, 2,1);
    add_edge(gr, 2,3);
    add_edge(gr, 3,1);
    add_edge(gr, 3,2);
    add_edge(gr, 4,1);
    add_edge(gr, 4,5);
    add_edge(gr, 4,8);
    add_edge(gr, 5,6);
    add_edge(gr, 5,7);
    add_edge(gr, 6,5);
    add_edge(gr, 6,8);
    add_edge(gr, 7,5);
    add_edge(gr, 7,6);
    add_edge(gr, 8,9);
    add_edge(gr, 9,10);

    bfs(gr, 0);
    return 0;
}
```