Breadth First Search
Graph Input- Adjacency Matrix

```
#include<stdio.h>

struct queue{
    int qe[20];
    int front,rear;
};
struct graph{
    int adj[20][20];
    int n;
};
void display(struct queue q){
    int i;
    for(i=q.front;i<q.rear;i++)
        printf("%d  ", q.qe[i]);
}
int qempty(struct queue q){
    if(q.front==-1 && q.rear==-1){
        return 1;
    }
    else
    {
        return 0;
    }

}
void insertrear(struct queue *q, int v){
    if(qempty(*q)){
        q->front++;
        q->rear++;
        q->qe[q->rear]=v;
        //printf("%dq ",q->qe[q->rear]);
```

```
    }
    else{
        q->rear++;
        q->qe[q->rear]=v;
    }
}
int deletefront( struct queue *q){
int i;
if(!qempty(*q)){
    i=q->qe[q->front];
    if(q->front==q->rear){
        q->front=-1;
        q->rear=-1;
    }
    else
        q->front++;
    return i;
}
else{
    return -1;
}
}
void bfs( struct graph g, int *visited, int v){
struct queue qu;
int i,j;
qu.front=-1;
qu.rear=-1;

insertrear(&qu,v);
visited[v]=1;
while(!qempty(qu)){
    i=deletefront(&qu);
    //visited[i]=1;
    printf("%d -->",i);
    for(j=0;j<g.n;j++){
```

```
        if(g.adj[i][j] && !visited[j]){
            insertrear(&qu,j);
            visited[j]=1;
            //display(qu);
        }
    }
    //printf("\n Queue Contents\n");
    //display(qu);

}

}
int main(){
    int i,j;
    int visited[20];
    struct graph G;
    printf("\n Enter the number of vertices in the graph : ");
    scanf("%d", &G.n);
    printf("\n Enter the adjacenecy matrix for graph G \n");
    for(i=0;i<G.n;i++){
        visited[i]=0;
        for(j=0;j<G.n;j++)
            scanf("%d", &G.adj[i][j]);
    }
    bfs(G,&visited,0);

}
```

Breadth First Search
Graph Input- Adjacency List

```
#include<stdio.h>
#include<stdlib.h>
struct node{
   int node_no;
   struct node *next;
};
struct graph{
   int n;
   struct node **nodelist;
};
struct queue{
   int qe[20];
   int front;
   int rear;
};
int qempty(struct queue q){
   if(q.front==-1 && q.rear==-1){
      return 1;
   }
   else
   {
      return 0;
   }

}
void qinsertrear(struct queue *q, int v){
   if(qempty(*q)){
      q->front++;
      q->rear++;
      q->qe[q->rear]=v;
      //printf("%dq ",q->qe[q->rear]);
```

```
    }
    else{
        q->rear++;
        q->qe[q->rear]=v;
    }
}
int qdeletefront( struct queue *q){
int i;
if(!qempty(*q)){
    i=q->qe[q->front];
    if(q->front==q->rear){
        q->front=-1;
        q->rear=-1;
    }
    else
        q->front++;
    return i;
}
else{
    return -1;
}
}
struct node *insertrear(struct node *nodelist, struct node *newnode){
    struct node *temp;
    if(nodelist==NULL){
        nodelist=newnode;
    }
    else{
        temp=nodelist;
        while(temp->next!=NULL)
            temp=temp->next;
        temp->next=newnode;
    }
    return nodelist;
}
```

```
void display(struct graph g){
struct node *temp;
int i;

for(i=0;i<g.n;i++){
   temp=g.nodelist[i];
      printf(" %d:-->",i);
      while(temp!=NULL){
         printf("%d -->",temp->node_no);
         temp=temp->next;
      }
      printf(" NULL\n ");
}
}
void bfs(struct graph *g, int *visited, int v){
   int i;
   struct node *temp;

   struct queue q;
   q.front=-1;
   q.rear=-1;
   qinsertrear(&q,v);
   visited[v]=1;
   while(!qempty(q)){
     i=qdeletefront(&q);
     temp=g->nodelist[i];

      printf("%d -->",i);
      while(temp!=NULL){
         if(!visited[temp->node_no]){
            qinsertrear(&q,temp->node_no);
            visited[temp->node_no]=1;

         }
         temp=temp->next;
```

```
        }
    }
printf("END");
}

int main(){
    int i, adjvertex,j,k;
    struct graph g;
    struct node *newnode;
    int *visited;
    printf("\n Enter the number of nodes in the graph: ");
    scanf("%d",&g.n);
    g.nodelist=(struct node **)malloc(g.n*sizeof(struct node *));
    visited=(int *)malloc(sizeof(g.n*sizeof(int)));
    for(i=0;i<g.n;i++)
        visited[i]=0;
    for(i=0;i<g.n;i++)
        g.nodelist[i]=NULL;
    for(i=0;i<g.n;i++){
        printf("\n Enter the number of nodes adjacent to %d: ",i);
        scanf("%d", &j);
        for(k=0;k<j;k++){
            printf("\n Enter the verticex adjacent to vertex %d.:  ",i);
            scanf("%d",&adjvertex);
            struct node *newnode=(struct node *)malloc(sizeof(struct
node));
            newnode->node_no=adjvertex;
            newnode->next=NULL;
            g.nodelist[i]=insertrear(g.nodelist[i],newnode);
        }
    }
    display(g);
        bfs(&g, visited,0);
    }
```