```
//adjacency matrix BFS
#include<stdio.h>
#include<stdlib.h>

#define MAX 100

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

int n;
int mat[MAX][MAX];
int state[MAX];
void accept();
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()
{
accept();
BF_Traversal();
return 0;
}

void accept()
{
int i, j;
char reply;
printf("How many vertices:");
scanf("%d",&n);
for ( i = 0 ; i < n ; i++ )
{
for ( j = 0 ; j < n ; j++ )
{

printf("\n Is there edge between %d & %d ? (Y/N/y/n) :",i,j);
scanf(" %c", &reply);
if ( reply == 'y' || reply == 'Y' )
mat[i][j] = 1;
else
```

```
mat[i][j] = 0;
}
}
}

void BF_Traversal()
{
int v;

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

printf("\nEnter Start Vertex for BFS: ");
scanf("%d", &v);
printf("\nBFS Traversal is: ");
BFS(v);
printf("\n\n");
}

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(mat[v][i] == 1 && state[i] == initial)
{
insert_queue(i);
state[i] = waiting;
}
}
}
printf("\n");
}
```

```
void insert_queue(int vertex)
{
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;
delete_item = queue[front];
front = front+1;
return delete_item;
}
```