```
#include <stdio.h>
int q[20], top = -1, front = -1, rear = -1, a[20][20],
vis[20], stack[20];
int delete ();
void add(int item);
void bfs(int s, int n);
void dfs(int s, int n);
void push(int item);
int pop();
int main()
{
    int n, i, s, ch, j;
    char c, dummy;
    printf("ENTER THE NUMBER VERTICES\t");
    scanf("%d", &n);
    for (i = 1; i <= n; i++)
    {
        for (j = 1; j <= n; j++)
        {
            printf("ENTER 1 IF %d HAS A NODE WITH %d ELSE
0\t", i, j);
            scanf("%d", &a[i][j]);
        }
    }
    printf("THE ADJACENCY MATRIX IS\n");
    for (i = 1; i <= n; i++)
    {
        for (j = 1; j <= n; j++)
        {
            printf(" %d", a[i][j]);
        }
        printf("\n");
    }
    do
    {
        for (i = 1; i <= n; i++)
            vis[i] = 0;
        printf("\nMENU");
        printf("\n1.B.F.S");
        printf("\n2.D.F.S");
        printf("\nENTER YOUR CHOICE ");

        scanf("%d", &ch);
        printf("ENTER THE SOURCE VERTEX : ");
```

```
        scanf("%d", &s);
        switch (ch)
        {
        case 1:
            bfs(s, n);
            break;
        case 2:
            dfs(s, n);
            break;
        }
        printf("DO U WANT TO CONTINUE(Y/N) ? ");
        scanf("%c", &dummy);
        scanf("%c", &c);
    } while ((c == 'y') || (c == 'Y'));
    return 0;
}
void bfs(int s, int n)
{
    int p, i;
    add(s);
    vis[s] = 1;
    p = delete ();
    if (p != 0)
        printf(" %d", p);
    while (p != 0)
    {
        for (i = 1; i <= n; i++)
            if ((a[p][i] != 0) && (vis[i] == 0))
            {
                add(i);
                vis[i] = 1;
            }
        p = delete ();
        if (p != 0)
            printf(" %d ", p);
    }
    for (i = 1; i <= n; i++)
        if (vis[i] == 0)
            bfs(i, n);
}
void add(int item)
{
    if (rear == 19)
        printf("QUEUE FULL");
```

```
    else
    {
        if (rear == -1)
        {
            q[++rear] = item;
            front++;
        }
        else
            q[++rear] = item;
    }
}
int delete ()
{
    int k;
    if ((front > rear) || (front == -1))
        return (0);
    else
    {
        k = q[front++];
        return (k);
    }
}
void dfs(int s, int n)
{
    int i, k;
    push(s);
    vis[s] = 1;
    k = pop();
    if (k != 0)
        printf(" %d ", k);
    while (k != 0)
    {
        for (i = 1; i <= n; i++)
            if ((a[k][i] != 0) && (vis[i] == 0))
            {
                push(i);
                vis[i] = 1;
            }
        k = pop();
        if (k != 0)
            printf(" %d ", k);
    }
    for (i = 1; i <= n; i++)
        if (vis[i] == 0)
```

```
            dfs(i, n);
}
void push(int item)
{
    if (top == 19)
        printf("Stack overflow ");
    else
        stack[++top] = item;
}
int pop()
{
    int k;
    if (top == -1)
        return (0);
    else
    {
        k = stack[top--];
        return (k);
    }
}
```