```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>

int n;

typedef struct queue
{
    int data;
    struct queue *next;
} q;

q *front = NULL;
q *ptr = NULL;
q *rear = NULL;

int emptyqueue()
{
    if (front)
    {
        return 0;
    }
    else
    {
        return 1;
    }
}
void enqueue(int key)
{
    ptr = (q *)malloc(sizeof(q));
    ptr->data = key;
    ptr->next = NULL;
    if (emptyqueue())
    {
        front = rear = ptr;
        return;
    }
    rear->next = ptr;
    rear = ptr;
}

int dequeue()
{
    if (!emptyqueue())
    {
        int key;
        ptr = front;
        key = front->data;
        front = front->next;
        free(ptr);
        return key;
    }
    return -1;
}
```

```c
int direction_graph(int graph[][n])
{
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++)
        {
            if (graph[i][j] == 1 && graph[j][i] == 1)
            {
                return 1;
            }
        }
    }

    return 0;
}

void bfs_traversal(int graph[][n], int s)
{
    int arr[n];
    memset(arr, 0, n * sizeof(arr[0]));
    enqueue(s);
    arr[s] = 1;
    printf("%d ", s);
    while (!emptyqueue())
    {
        int x = dequeue();
        for (int i = 0; i < n; i++)
        {
            if (arr[i] == 0 && graph[x][i] == 1)
            {
                arr[i] = 1;
                printf("%d ", i);
                enqueue(i);
            }
        }
    }
}

void dfs_traversal(int graph[][n], int s, int arr[])
{
    if (arr[s] == 0)
    {
        arr[s] = 1;
        printf("%d ", s);
        for (int i = 0; i < n; i++)
        {
            if (!arr[i] && graph[s][i] == 1)
            {
                dfs_traversal(graph, i, arr);
            }
        }
    }
}
void FreeAll()
{

    free(rear);
    free(front);
    free(ptr);
}

int main()
{

    printf("Enter the number of edges\n");
    scanf("%d", &n);
    int graph[n][n];
    int arr[n];
    printf("Enter the edges Elements\n");
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++)
        {
            scanf("%d", &graph[i][j]);
        }
    }
    while (1)
    {
        printf("Enter you task to perform\n");
        printf("1. Type of Grammar  2. BFS  3.DFS  4. To Exit\n");
        int op, s;
        scanf("%d", &op);
        switch (op)
        {
        case 1:
            if (direction_graph(graph))
            {
                printf("The Graph is Undirected\n");
            }
            else
            {
                printf("The graph is Directed\n");
            }
            break;
        case 2:
            printf("BFS Traversal\n");
            printf("Enter the element from where you want to start the
path\n");
            scanf("%d", &s);
            bfs_traversal(graph, s);
            break;
        case 3:
            printf("DFS Traversal\n");
            printf("Enter the element from where you want to start the
path\n");
            scanf("%d", &s);

            memset(arr, 0, n * sizeof(arr[0]));
            dfs_traversal(graph, s, arr);
            break;
        case 4:
            exit(0);
            break;
        default:
            printf("Please Enter the correct task number\n");
        }
    }
    return 0;
}
```