PALAK SHARMA
73

```
#include <stdio.h>
#include <stdlib.h>


int vis[100];


struct Graph {
	int V;
	int E;
	int** Adj;
};


struct Graph* adjMatrix()
{
	struct Graph* G = (struct Graph*)
		malloc(sizeof(struct Graph));
	if (!G) {
		printf("Memory Error\n");
		return NULL;
	}
	G->V = 7;
	G->E = 7;

	G->Adj = (int**)malloc((G->V) * sizeof(int*));
	for (int k = 0; k < G->V; k++) {
		G->Adj[k] = (int*)malloc((G->V) * sizeof(int));
	}

	for (int u = 0; u < G->V; u++) {
		for (int v = 0; v < G->V; v++) {
			G->Adj[u][v] = 0;
		}
	}
	G->Adj[0][1] = G->Adj[1][0] = 1;
	G->Adj[0][2] = G->Adj[2][0] = 1;
	G->Adj[1][3] = G->Adj[3][1] = 1;
	G->Adj[1][4] = G->Adj[4][1] = 1;
	G->Adj[1][5] = G->Adj[5][1] = 1;
	G->Adj[1][6] = G->Adj[6][1] = 1;
	G->Adj[6][2] = G->Adj[2][6] = 1;

	return G;
}

// DFS function to print DFS traversal of graph
void DFS(struct Graph* G, int u)
{
	vis[u] = 1;
	printf("%d ", u);
	for (int v = 0; v < G->V; v++) {
		if (!vis[v] && G->Adj[u][v]) {
			DFS(G, v);
		}
	}
```

```
}


void DFStraversal(struct Graph* G)
{
	for (int i = 0; i < 100; i++) {
		vis[i] = 0;
	}
	for (int i = 0; i < G->V; i++) {
		if (!vis[i]) {
			DFS(G, i);
		}
	}
}


void main()
{
	struct Graph* G;
	G = adjMatrix();
	DFStraversal(G);
}
```