```
//Created By Ritwik Chandra Pandey
//On 4th Nov
//Implementing a directed graph and its operations using adjacency matrix


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


int graph [20][20];

void print(int * N) {
	int i,j;
	if(*N==0){
		printf("Graph is empty.\n");
		return;
	}
	for(i=1;i<=*N;i++){
		for(j=1;j<=*N;j++){
			printf("%d\t",graph[i][j]);
		}
		printf("\n");
	}
}

void insertVertex(int * N) {
	int x[10],y[10];
	int t,i,s;
	*N = *N + 1;
	printf("Enter the number edges from the existing vertices to new vertex : ");
	scanf("%d",&s);
	printf("Enter the source of each edge : ");
	for(i=1;i<=s;i++){
		scanf("%d",&x[i]);

	}
	printf("Enter the number edges from the new vertex to existing vertices : ");
	scanf("%d",&t);
	printf("Enter the destination of each edge : ");
	for(i=1;i<=t;i++){
```

```
                scanf("%d",&y[i]);
        }
        for(i=1;i<=*N;i++){
                graph[i][*N] = 0;
                graph[*N][i] = 0;
        }
        for(i=1;i<=s;i++){
                if(x[i]<*N){
                        graph[x[i]][*N] = 1;
                }
                else{
                        printf("Invalid vertex.\n");
                }
        }
        for(i=1;i<=t;i++){
                if(y[i]<=*N){
                        graph[*N][y[x[i]]] = 1;
                }
                else{
                        printf("Invalid vertex.\n");
                }
        }
        printf("After inserting vertex the adjacency matrix is : \n");
        print(N);
}

void insertEdge(int *N) {
        int v1, v2;
        printf("Enter the source vertex of the edge : ");

        scanf("%d",&v1);
        printf("Enter the destination vertex of the edge : ");
        scanf("%d",&v2);
        if(v1<=*N && v2<=*N){
                graph[v1][v2] = 1;

        }else{
                printf("Invalid vertex.\n");
                return;
        }
```

```
	printf("After inserting edge the adjacency matrix is : \n");
	print(N);
}

void deleteVertex(int *N) {
	int vd, i, j, k;
	if(*N == 0){
		printf("Graph is empty.\n");
		return;
	}
	printf("Enter the vertex to be deleted : ");
	scanf("%d",&vd);
	if(vd>*N){
		printf("Invalid vertex.\n");
		return;
	}
	j = vd;
	for(i=j; i<=*N-1;i++){
		for(k = 1; k<=*N;k++){
			graph[i+1][k] = graph[i][k];
		}
	}
	for(i=j;i<=*N-1;i++){
		for(k=1;k<=*N;k++){
			graph[i][k] = graph[i+1][k];
		}
	}
	*N = *N - 1;
	printf("After deleting vertex the adjacency matrix is : \n");
	print(N);
}

void deleteEdge(int *N) {
	int v1,v2;
	printf("Enter the source vertex of the edge : ");
	scanf("%d",&v1);
	printf("Enter the destination vertex of the edge : ");
	scanf("%d",&v2);
	if(v1<=*N && v2<=*N){
		if(graph[v1][v2]==0){
				printf("Edge does not exist.\n");
				return;
			}
			graph[v1][v2] = 0;
		}else{
			printf("Invalid vertex.\n");
			return;
		}
		printf("After deleting edge the adjacency matrix is : \n");
		print(N);
}

void main() {
		int x, op;
		int N,E,s,d,i,j;
		printf("Enter the number of vertices : ");
		scanf("%d",&N);
		printf("Enter the number of edges : ");
		scanf("%d",&E);
		for(i=1;i<=E;i++) {
			printf("Enter source : ");
			scanf("%d",&s);
			printf("Enter destination : ");
			scanf("%d",&d);
			if(s > N || d > N || s<=0 || d<=0) {
				printf("Invalid index. Try again.\n");
				i--;
				continue;
			} else {
				graph[s][d] = 1;
			}
		}
		while(1)
		{
			printf("1.Insert vertex 2.Insert edge 3.Delete vertex 4.Delete edge 5.Print adjacency matrix 6.Exit\n");
			printf("Enter your option : ");
			scanf("%d", &op);
			switch(op) {
				case 1:
						insertVertex(&N);
```

```
                        break;
            case 2:
                        insertEdge(&N);
                        break;
            case 3:
                        deleteVertex(&N);
                        break;
            case 4:
                        deleteEdge(&N);
                        break;
            case 5:
                        print(&N);
                        break;
            case 6:
                        exit(0);
        }
    }
}
```