

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

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

```
struct node {
    struct node *next;
    int vertex;
};
typedef struct node *GNODE;
GNODE graph[20];
```

```
void print(int * N) {
    int i=0;
    GNODE p;
    for(i=1;i<=*N;i++){
        p = graph[i];

        if(p!=NULL){
            printf("%d=>",i);
        }else{
            continue;
        }while(p!=NULL){
            printf("%d\t",p->vertex);
            p = p->next;
        }
        printf("\n");
    }
}
```

```
void insertVertex(int * N) {
    int x[10],y[10];
    int s,t,d,i;
    GNODE p,q;
    *N++;
    printf("Enter the number edges from the existing vertices to new vertex : ");
```

```

scanf("%d",&s);
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);
for(i=1;i<=s;i++){
    scanf("%d",&y[i]);
}
for(i=1;i<=s;i++){
    if(x[i]<=N){
        q = malloc(sizeof(GNODE));
        q->vertex = x[i];
        q->next = NULL;

        if(graph[x[i]]==NULL){
            graph[x[i]] = q;
        }else{
            p = graph[x[i]];
            while(p->next!=NULL){
                p = p->next;
            }
            p->next = q;
        }
    }
    else{
        printf("Invalid vertex.\n");
    }
}
for(i=1;i<=t;i++){
    if(y[i]<=N){
        q = malloc(sizeof(GNODE));
        q->vertex = y[i];
        q->next = NULL;
        if(graph[*N] == NULL){
            graph[*N] = q;
        }
        else{
            p = graph[*N];
            while(p->next!=NULL){

```

```

                p = p->next;
            }
            p->next = q;
        }
    }
    else{
        printf("Invalid vertex.\n");
    }
}
printf("After inserting vertex the adjacency list is : \n");
print(N);
}

```

```

void insertEdge(int *N) {
    int v1,v2;
    GNODE p,q;
    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){
        q = malloc(sizeof(GNODE));
        q->vertex = v2;

        q->next=NULL;
        if(graph[v1]==NULL){
            graph[v1] = p;
        }
        else{
            p = graph[v1];
            while(p->next!=NULL){
                p = p->next;
            }
            p->next=q;
        }
    }
    else{
        printf("Invalid vertex.\n");
        return;
    }
}

```

```

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

void deleteVertex(int *N) {
    int vd,i,j,k;
    GNODE temp, prev;
    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;
    }
    graph[vd] = NULL;
    *N = *N -1;
    for(i = 1;i<=*N;i++){
        temp = graph[i];
        if(temp!=NULL && temp->vertex== vd){
            graph[i]=temp->next;
            free(temp);
            continue;
        }
        while(temp!=NULL && temp->vertex!= vd){
            prev =temp;
            temp = temp->next;
        }
        if(temp!=NULL && temp->vertex== vd){
            prev->next = temp->next;
            free(temp);
        }
    }

    printf("After deleting vertex the adjacency list is : \n");
    print(N);
}

```

```

}

void deleteEdge(int *N) {
    int v1,v2;
    GNODE temp,prev;
    printf("Enter the source vertex of the edge : ");
    scanf("%d",&v1);
    printf("Enter the destination vertex of the edge : ");
    scanf("%d",&v2);
    temp = graph[v1];
    if(temp->vertex == v2){
        graph[v1] = temp->next;
        free(temp);
    }
    while(temp!=NULL && temp->vertex!=v2){
        prev = temp;
        temp= temp->next;
    }
    if(temp->vertex==v2){
        prev->next = temp->next;
        free(temp);
    }
    printf("After deleting edge the adjacency list is : \n");
    print(N);
}

```

```

void main() {
    int x, op;
    int N,E,s,d,i,j;
    GNODE p,q;
    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);
        q=(GNODE)malloc(sizeof(struct node));
    }
}

```

```

        q->vertex=d;
        q->next=NULL;
        if(graph[s]==NULL)
            graph[s]=q;
        else {
            p=graph[s];
            while(p->next!=NULL)
                p=p->next;
            p->next=q;
        }
    }
    while(1) {
        printf("1.Insert vertex 2.Insert edge 3.Delete vertex 4.Delete edge 5.Print adjacency list 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);
        }
    }
}

```