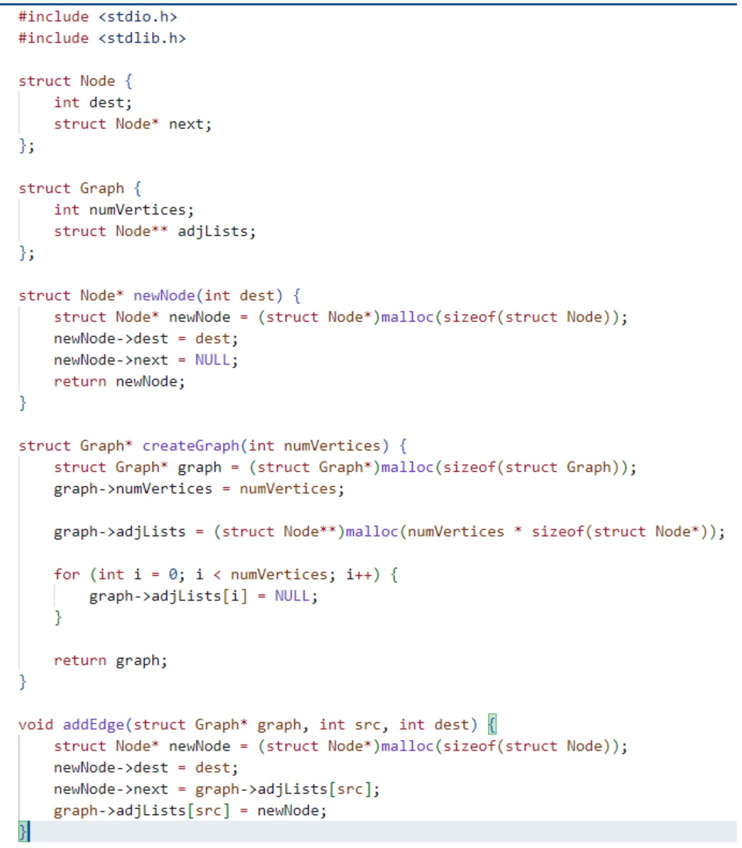
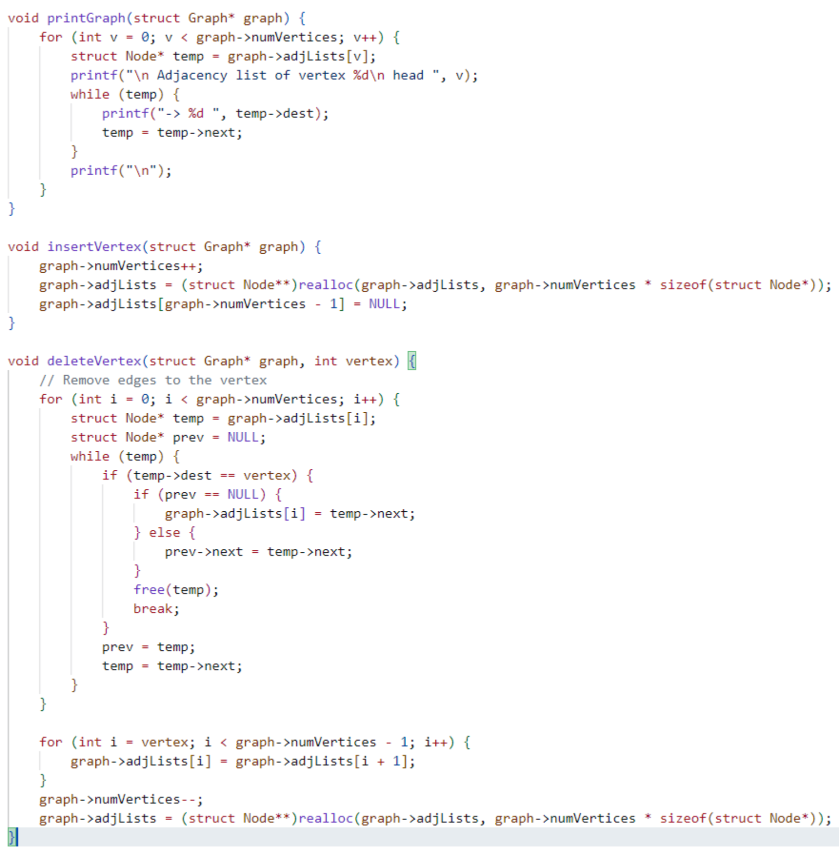
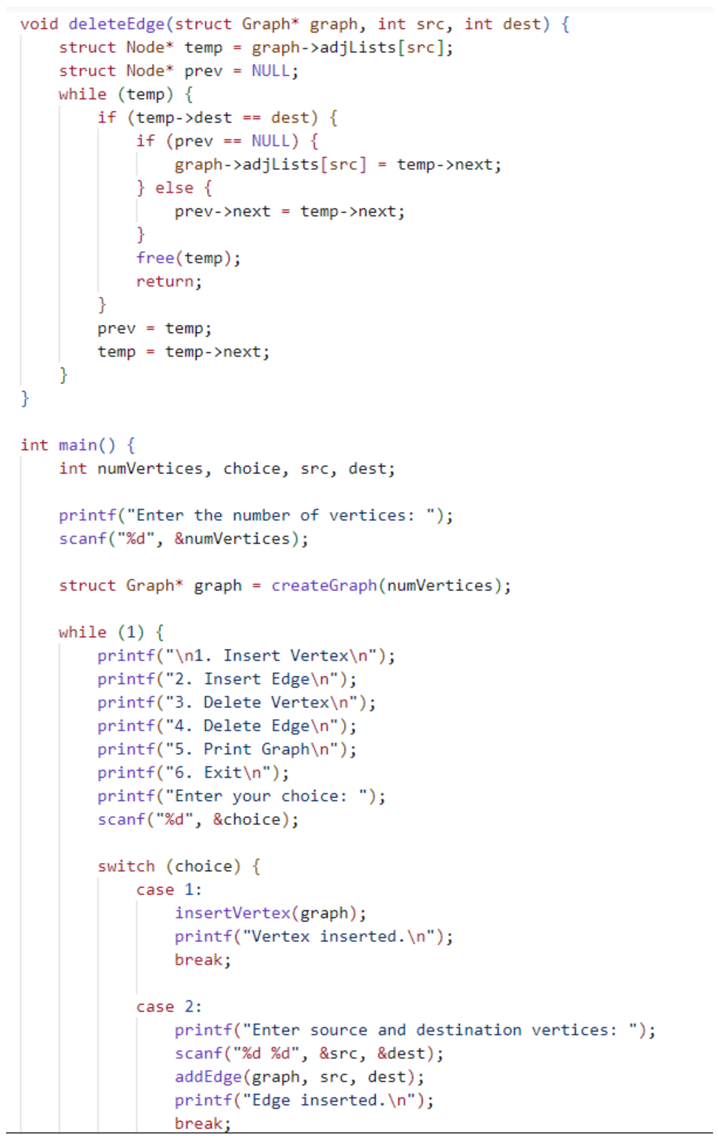


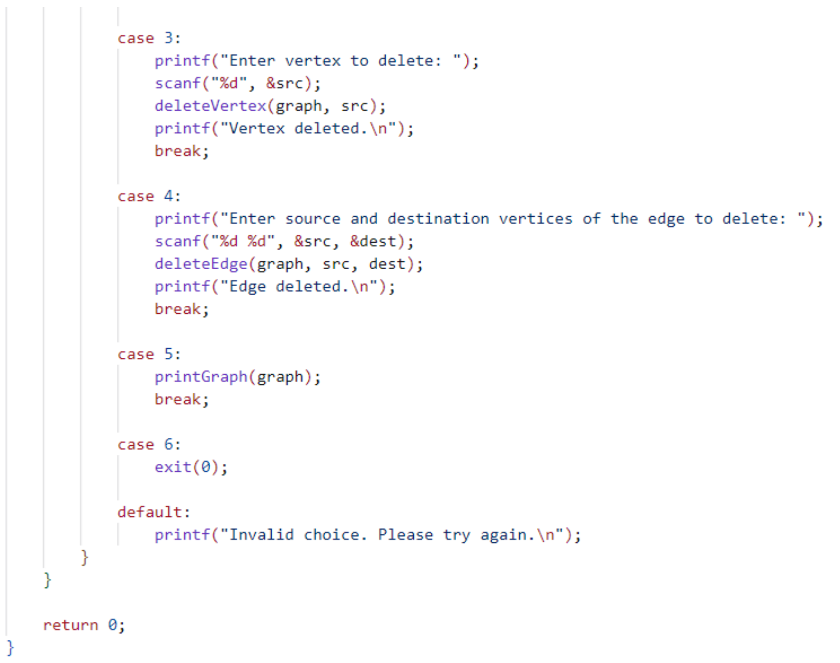
```c
#include <stdio.h>
#include <stdlib.h>

struct Node {
    int dest;
    struct Node* next;
};

struct Graph {
    int numVertices;
    struct Node** adjLists;
};

struct Node* newNode(int dest) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->dest = dest;
    newNode->next = NULL;
    return newNode;
}

struct Graph* createGraph(int numVertices) {
    struct Graph* graph = (struct Graph*)malloc(sizeof(struct Graph));
    graph->numVertices = numVertices;

    graph->adjLists = (struct Node**)malloc(numVertices * sizeof(struct Node*));

    for (int i = 0; i < numVertices; i++) {
        graph->adjLists[i] = NULL;
    }

    return graph;
}

void addEdge(struct Graph* graph, int src, int dest) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->dest = dest;
    newNode->next = graph->adjLists[src];
    graph->adjLists[src] = newNode;
}
```



```c
void printGraph(struct Graph* graph) {
    for (int v = 0; v < graph->numVertices; v++) {
        struct Node* temp = graph->adjLists[v];
        printf("\n Adjacency list of vertex %d\n head ", v);
        while (temp) {
            printf("-> %d ", temp->dest);
            temp = temp->next;
        }
        printf("\n");
    }
}

void insertVertex(struct Graph* graph) {
    graph->numVertices++;
    graph->adjLists = (struct Node**)realloc(graph->adjLists, graph->numVertices * sizeof(struct Node*));
    graph->adjLists[graph->numVertices - 1] = NULL;
}

void deleteVertex(struct Graph* graph, int vertex) {
    // Remove edges to the vertex
    for (int i = 0; i < graph->numVertices; i++) {
        struct Node* temp = graph->adjLists[i];
        struct Node* prev = NULL;
        while (temp) {
            if (temp->dest == vertex) {
                if (prev == NULL) {
                    graph->adjLists[i] = temp->next;
                } else {
                    prev->next = temp->next;
                }
                free(temp);
                break;
            }
            prev = temp;
            temp = temp->next;
        }
    }

    for (int i = vertex; i < graph->numVertices - 1; i++) {
        graph->adjLists[i] = graph->adjLists[i + 1];
    }
    graph->numVertices--;
    graph->adjLists = (struct Node**)realloc(graph->adjLists, graph->numVertices * sizeof(struct Node*));
}
```



```c
void deleteEdge(struct Graph* graph, int src, int dest) {
    struct Node* temp = graph->adjLists[src];
    struct Node* prev = NULL;
    while (temp) {
        if (temp->dest == dest) {
            if (prev == NULL) {
                graph->adjLists[src] = temp->next;
            } else {
                prev->next = temp->next;
            }
            free(temp);
            return;
        }
        prev = temp;
        temp = temp->next;
    }
}

int main() {
    int numVertices, choice, src, dest;

    printf("Enter the number of vertices: ");
    scanf("%d", &numVertices);

    struct Graph* graph = createGraph(numVertices);

    while (1) {
        printf("\n1. Insert Vertex\n");
        printf("2. Insert Edge\n");
        printf("3. Delete Vertex\n");
        printf("4. Delete Edge\n");
        printf("5. Print Graph\n");
        printf("6. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                insertVertex(graph);
                printf("Vertex inserted.\n");
                break;

            case 2:
                printf("Enter source and destination vertices: ");
                scanf("%d %d", &src, &dest);
                addEdge(graph, src, dest);
                printf("Edge inserted.\n");
                break;
```



```c
            case 3:
                printf("Enter vertex to delete: ");
                scanf("%d", &src);
                deleteVertex(graph, src);
                printf("Vertex deleted.\n");
                break;

            case 4:
                printf("Enter source and destination vertices of the edge to delete: ");
                scanf("%d %d", &src, &dest);
                deleteEdge(graph, src, dest);
                printf("Edge deleted.\n");
                break;

            case 5:
                printGraph(graph);
                break;

            case 6:
                exit(0);

            default:
                printf("Invalid choice. Please try again.\n");
        }
    }

    return 0;
}
```