exp_9/a.c

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

#define MAX_VERTICES 100

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

struct Graph {
    int numVertices;
    struct Node* adjLists[MAX_VERTICES];
    int visited[MAX_VERTICES];
};

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

struct Graph* createGraph(int vertices) {
    struct Graph* graph = (struct Graph*) malloc(sizeof(struct Graph));
    graph->numVertices = vertices;
    for (int i = 0; i < vertices; i++) {
        graph->adjLists[i] = NULL;
        graph->visited[i] = 0;
    }
    return graph;
}

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

    // For undirected graph, add edge from dest to src too
    newNode = createNode(src);
    newNode->next = graph->adjLists[dest];
    graph->adjLists[dest] = newNode;
}

void DFS(struct Graph* graph, int vertex) {
    struct Node* adjList = graph->adjLists[vertex];
    struct Node* temp = adjList;

```

```c
    graph->visited[vertex] = 1;
    printf("%d ", vertex);

    while (temp != NULL) {
        int connectedVertex = temp->vertex;
        if (graph->visited[connectedVertex] == 0) {
            DFS(graph, connectedVertex);
        }
        temp = temp->next;
    }
}

void BFS(struct Graph* graph, int startVertex) {
    int queue[MAX_VERTICES];
    int front = 0, rear = 0;

    graph->visited[startVertex] = 1;
    queue[rear++] = startVertex;

    while (front != rear) {
        int currentVertex = queue[front++];
        printf("%d ", currentVertex);
        struct Node* temp = graph->adjLists[currentVertex];

        while (temp) {
            int adjVertex = temp->vertex;
            if (graph->visited[adjVertex] == 0) {
                graph->visited[adjVertex] = 1;
                queue[rear++] = adjVertex;
            }
            temp = temp->next;
        }
    }
}

void resetVisited(struct Graph* graph) {
    for (int i = 0; i < graph->numVertices; i++) {
        graph->visited[i] = 0;
    }
}

int main() {
    int vertices, edges, u, v, choice, startVertex;
    printf("Enter number of vertices: ");
    scanf("%d", &vertices);
    struct Graph* graph = createGraph(vertices);

    printf("Enter number of edges: ");
    scanf("%d", &edges);
    printf("Enter edges (source destination):\n");
```

```c
    for (int i = 0; i < edges; i++) {
        scanf("%d %d", &u, &v);
        addEdge(graph, u, v);
    }

    while (1) {
        printf("\n1: BFS\n2: DFS\n0: Exit\nEnter choice: ");
        scanf("%d", &choice);
        if (choice == 0) break;
        printf("Enter starting vertex: ");
        scanf("%d", &startVertex);

        resetVisited(graph);

        if (choice == 1) {
            printf("BFS traversal: ");
            BFS(graph, startVertex);
            printf("\n");
        } else if (choice == 2) {
            printf("DFS traversal: ");
            DFS(graph, startVertex);
            printf("\n");
        } else {
            printf("Invalid choice\n");
        }
    }

    free(graph);
    return 0;
}

```