Institute for Advancing Intelligence, TCG CREST
(TCG Centres for Research and Education in Science and Technology)

Introduction to Programming and Data Structures
Ph.D. Coursework: First year, First Semester (Session: 2024-25)
Homework #03

Full Marks: 00 | Instructor: Dr. Laltu Sardar
Clarification Deadline: **2024-Nov-18** | Submission Deadline: **2024-Nov-19**

Instructions:

1. Keep this assignment ready before the next class.

Problem Statement

Given a graph $G = (V, E)$, where V is the set of vertices and E is the set of edges, the input specifies the number of vertices n and the edges in the graph. The task is to implement a graph structure using an adjacency list represented by linked lists and to write functions for reading, initializing, updating, and querying the graph.

Input File Format

The input file contains the following:

- The first line contains n, the number of vertices in the graph.
- Each subsequent line contains two integers a_i and b_i separated by a space, representing an edge between vertices a_i and b_i.

Tasks and Implementation

1. Graph Structure Definition

Define the graph using the `typedef` structure with the following fields:

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

typedef struct graph_t {
        int n;          // Number of vertices
        Node **AdjLists; // Array of pointers to adjacency lists
} Graph;
```

2. Functions to Implement

1. **Read `n` from the input file**: Write a function to extract the number of vertices n from the input file.

```
int read_n(FILE *file);
```

2. **Initialize the Graph**: Allocate memory for the adjacency list array.

```
Graph* init_graph(int n);
```

3. **Add Edges to the Graph**: Insert edges into the adjacency list using a function.

```
void add_edge(Graph *graph, int a, int b);
```

4. **Read Edges from File**: Read all edges from the file and populate the adjacency list using `add_edge`.

```
void read_edges(Graph *graph, FILE *file);
```

5. **Display the Graph**: Print the adjacency list representation of the graph.

```
void display_graph(Graph *graph);
```

6. **Display Neighbors of a Vertex**: Given a vertex k, print all its neighbors.

```
void display_neighbor(Graph *graph, int k);
```

Example Input and Output

Input File (`graph_input.txt`):

```
5
0 1
0 2
1 2
1 3
3 4
```

Output:

Adjacency List:

```
Vertex 0: 1 -> 2
Vertex 1: 0 -> 2 -> 3
Vertex 2: 0 -> 1
Vertex 3: 1 -> 4
Vertex 4: 3
```

Neighbors of Vertex $k = 1$:

```
Neighbors of Vertex 1: 0, 2, 3
```