Data Structure Lab
Lab 15
Graphs Implemetation

Task 1 : Implement a Graph Data Structure using a 2D Array in C++

Description:
In this lab task, you will implement a graph data structure using a 2D array. The program should allow the user to create a graph, add edges and vertices, and perform various operations on the graph, such as traversal and finding the shortest path.
Instructions:

1. Implement a class for the graph data structure using a 2D array representation.
2. The class should have functions for adding vertices, adding edges, and performing graph traversals.

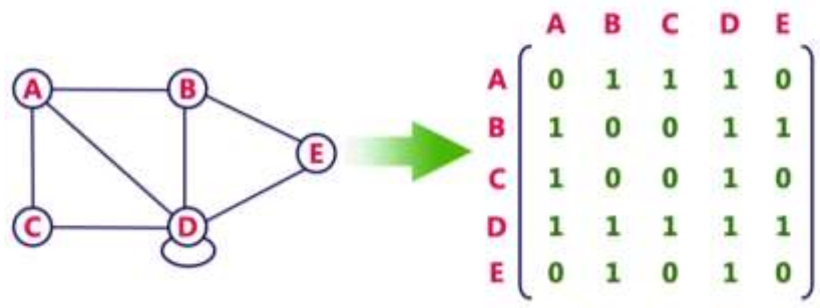
.
Here are the necessary functions that the class should implement:

1. **Graph(int n)**: A constructor that creates an empty graph with n vertices.
2. **void addEdge(int from, int to, int weight)**: Adds an edge from vertex "from" to vertex "to" with a weight.
3. **void addVertex()**: Adds a new vertex to the graph.
4. **void printGraph():** Prints the graph.

```
class Graph {

    bool** adjMatrix;
    int numVertices;

    Graph(int numVertices);             //constructor to set size of matrix
    void addEdge(int i, int j);         //marks the edge true in matrix
    void removeEdge(int i, int j);  //removes the edge at provided position
    bool isEdge(int i, int j);          //checks if edge exits or not, then returns
    void print();                       // prints the matrix
}
```



$$\begin{array}{c|ccccc} & A & B & C & D & E \\ \hline A & 0 & 1 & 1 & 1 & 0 \\ B & 1 & 0 & 0 & 1 & 1 \\ C & 1 & 0 & 0 & 1 & 0 \\ D & 1 & 1 & 1 & 1 & 1 \\ E & 0 & 1 & 0 & 1 & 0 \end{array}$$

Task 2 : Implement a Graph Data Structure using a Node-based Approach (Adjacency list) in C++

In this task, you will be implementing a graph data structure using a node-based implementation with an adjacency list.

To get started, create a **GraphNode** class that represents a node in the graph. This class should contain the following:

```
class GraphNode
{
public:
   int data;
   next pointer
};
```

Next, create a **Graphh** class that represents the entire graph. This class should contain the following:

```
class Graphh
{
private:
   int maxVertices;
   GraphNode** adjacencyList;

public:

   Graphh()
   // Constructor to create an empty graph with n vertices
     // allocate memory for the 2D array
  // Initialize all elements to false (no edges)
   void insertEdge(int source, int destination) //adds an edge between two nodes
```

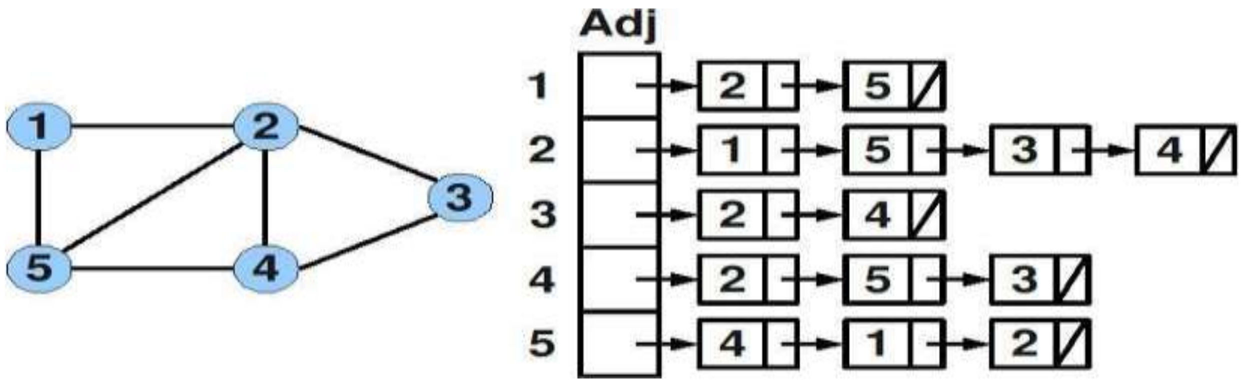
```
    void display()   // prints out the adjacency list representation of the graph

    ~Graphh()    //destructor to free allocated memory

};
```

Structure

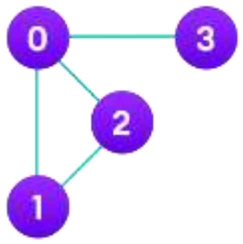
The relationship between the vertices in a graph is expressed using a set of directed/undirected Edges, where each edge connects one pair of vertices. Here is an example of undirected graph and its representation in the form of an adjacency list.



Task 3:

While connecting different cities, cycles can lead to inefficient routes and delays. Your task is to find the number of cycles in a given Undirected Graph by using the Depth-First Search (DFS) algorithms.

Example:



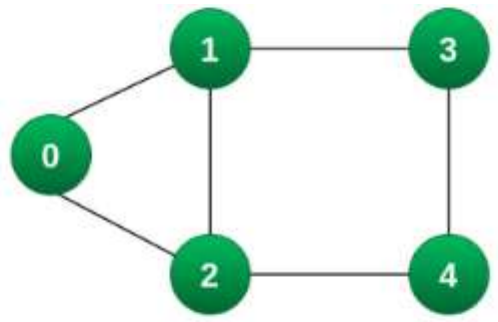
Output:

Number of cycles: 1 that is 0, 1, 2.

Task 4 (Bonus):

Breadth-First Search (BFS) is another graph traversal algorithm that explores a graph level by level. It starts at the source node and explores all the neighboring nodes at the present depth before moving on to nodes at the next level of depth.

Example:



Output: The start node is 0, then the **BFS traversal → 0-1-2-3-4**