Quiz No. <10.1>	
<Graph>	
Course Code: CPE010	**Program: BSCPE**
Course Title: Data Structures and Algorithms	**Date Performed: 10/2/2025**
Section: CPE21S4	**Date Submitted: 10/2/2025**
Name(s): Anastacio, Lester Arvid P.	**Instructor: Engr. Jimlord Quejado**

6. Output

7. Supplementary Activity

I. **Define the following**

A. Graph

- A graph is a data structure consisting of a set of vertices which are the nodes and edges that connect pairs of vertices. It models relationships between objects and this is something we see often as it is used to many things and one example of this is the ever growing internet wherein we can use it to look into maps of the world to find the quickest and shortest routes, it is an graph since it shows the map where you can see cities and roads that will help you traverse or guide you to path quicker to take you to your destination.

B. Vertex

- This is a fundamental unit in a graph in which represents an object or entity and these units are also connected to edges, you could say these are the building blocks of the graph as this connects the pairs of edges which is quite fairly important for the graph.

C. Nodes

- These are points in which is also called vertices, these can be found in mostly structures as it is used to sort things out within the graph or as a guide for both the edges as it is the meeting point of edges.

D. Weight

- This is a numerical value assigned to an edge in a graph, often representing cost, distance, or time between two nodes, it's like a way to determine the distance and cost that two edges needed for it to meet.

E. Path

- A path is a sequence of edges that connects a series of vertices in a graph to show there is a relationship between them this shows how to travel from one node to another it represents a relationship or route between connected points.

F. Directed graph

- This is a graph where edges have a direction, meaning they go from one vertex to another in a specific way in which a better example of this is moving from the letter A to B, this means that the relationship between the nodes is one-way.

G. Connected graph

- A connected graph is one in which there is a path between every pair of vertices. This means no node is isolated, and the graph forms a single unified structure these graphs are crucial in ensuring communication or flow is possible throughout the system.

H. Directed cyclic

- A directed cyclic graph is a directed graph that contains at least one cycle it is a path that starts and ends at the same vertex, following the direction of edges

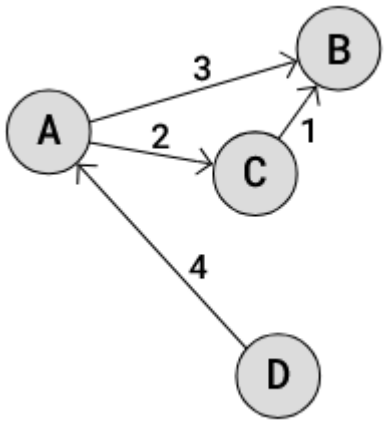
I. Adjacency matrix

- An adjacency matrix is a way of representing a graph using a 2D array or matrix and each row and column corresponds to a vertex, and the value at position indicates whether there is an edge from vertex A to vertex B.

J. Adjacency List

- In an adjacency list is a space-efficient way to represent a graph, especially when the graph is sparse which means it has a relatively few edges compared to the number of vertices, in this structure, each vertex maintains a list of all the vertices it is directly connected to. These lists can be implemented using arrays, linked lists, or other dynamic data structures.

II. Identify the following parts:



1. Vertices (A, B, C, D)
2. Edges:

D → A
A → C
A → B
C → B

III. Plot the adjacency matrix of the graph above.

	A	B	C	D
A	0	3	2	0
B	0	0	0	0
C	0	1	0	0
D	4	0	0	0

IV. Create a program in C++ program to create a graph that looks like in the above figure.

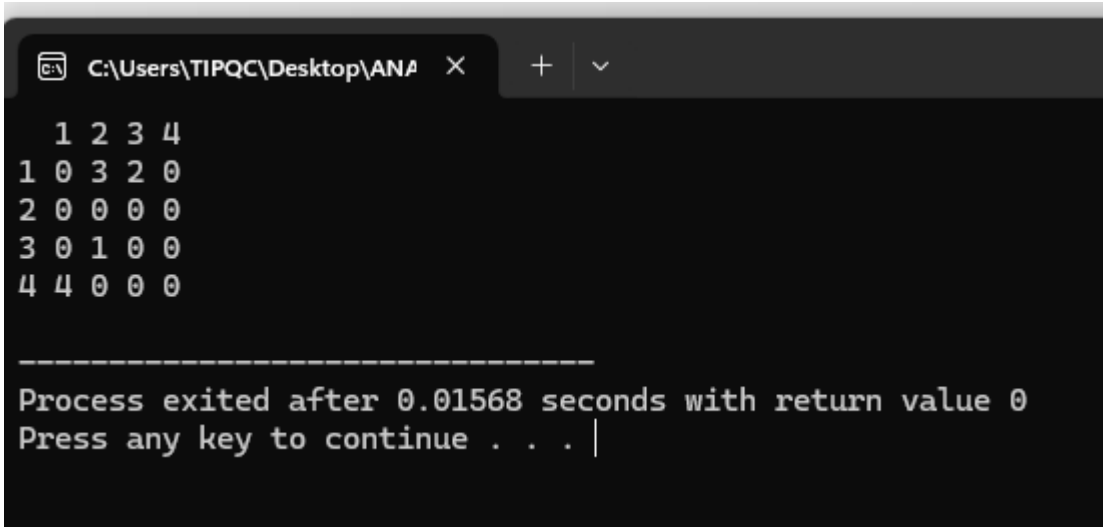
CODE:

[*] Graph.cpp

```cpp
#include <iostream>
#include <map>

class Vertex {
public:
    int id;
    std::map<int, int> connectedTo;

    Vertex() {}
    Vertex(int key) {
        id = key;
    }
    void addNeighbor(int nbr, int weight = 0) {
        connectedTo[nbr] = weight;
    }
    int getId() {
        return id;
    }
    int getWeight(int nbr) {
        std::map<int, int>::iterator it = connectedTo.find(nbr);
        if (it != connectedTo.end()) {
            return it->second;
        }
        return 0;
    }
    std::map<int, int> getConnections() {
        return connectedTo;
    }
};

class Graph {
public:
    std::map<int, Vertex> vertList;
    int numVertices;

    Graph() {
        numVertices = 0;
    }
```

```
    }
    void addVertex(int key) {
        numVertices++;
        Vertex newVertex(key);
        vertList[key] = newVertex;
    }
    bool contains(int n) {
        return vertList.find(n) != vertList.end();
    }
    void addEdge(int f, int t, int cost = 0) {
        if (!contains(f)) {
            std::cout << f << " was not found, adding!" << std::endl;
            addVertex(f);
        }
        if (!contains(t)) {
            std::cout << t << " was not found, adding!" << std::endl;
            addVertex(t);
        }
        vertList[f].addNeighbor(t, cost);
    }
    void printAdjacencyMatrix() {
        int keys[100];
        int count = 0;
        for (std::map<int, Vertex>::iterator it = vertList.begin(); it != vertList.end(); ++it) {
            keys[count++] = it->first;
        }

        std::cout << "  ";
        for (int i = 0; i < count; i++) {
            std::cout << keys[i] << " ";
        }
        std::cout << std::endl;

        for (int i = 0; i < count; i++) {
            std::cout << keys[i] << " ";
            for (int j = 0; j < count; j++) {
                std::cout << vertList[keys[i]].getWeight(keys[j]) << " ";
            }
            std::cout << std::endl;
        }
    }
};

int main() {
    Graph graph;

    graph.addVertex(1);  // A
    graph.addVertex(2);  // B
    graph.addVertex(3);  // C
    graph.addVertex(4);  // D

    graph.addEdge(1, 2, 3); // A -> B (3)
    graph.addEdge(1, 3, 2); // A -> C (2)
    graph.addEdge(3, 2, 1); // C -> B (1)
    graph.addEdge(4, 1, 4); // D -> A (4)

    graph.printAdjacencyMatrix();

    return 0;
}
```

OUTPUT:



V. Differentiate a BFS vs DFS types of search.

8. Conclusion

In this quiz, I learned a lot about graphs,edges and vertices and the ways how to use it further, I also struggled a lot at the part of the code, since im not that familiar with the graphs yet but it is still a nice experience.

9. Assessment Rubric