

GitHub link - <https://github.com/LasithaJananjaya/CS2023-Data-Structures-and-Algorithms---Workspace/tree/main/in-class-lab10>

Section 1 - Implementing Graph ADT

1. Write the adjacency list representation for the graph in Fig1.

1 -> 2 -> 3 -> 4 -> 5

2 -> 1 -> 3 -> 6

3 -> 1 -> 2

4 -> 1 -> 6 -> 7 -> 8

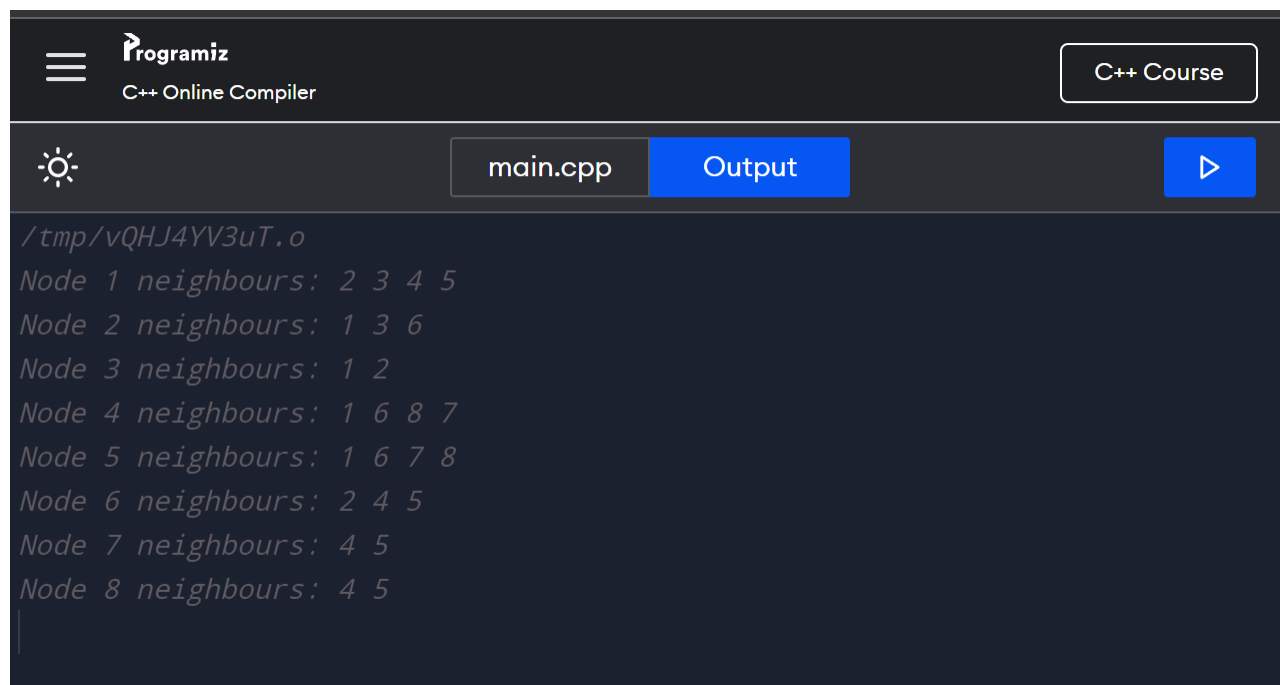
5 -> 1 -> 6 -> 7 -> 8

6 -> 2 -> 4 -> 5

7 -> 4 -> 5

8 -> 4 -> 5

2. By using comments provided in the code, complete the following *Node (struct)*, *intializenodes*, *addedge*, *print* functions.
3. Create graph object and add the graph in Fig.1.
4. Print the adjacency list using the print function you implemented and take screenshot.



The screenshot shows the Programiz C++ Online Compiler interface. The top bar includes the Programiz logo, "C++ Online Compiler", and a "C++ Course" button. Below the bar, there are tabs for "main.cpp" and "Output", with a play button to the right. The "Output" tab is active, displaying the following text:

```
/tmp/vQHJ4YV3uT.o
Node 1 neighbours: 2 3 4 5
Node 2 neighbours: 1 3 6
Node 3 neighbours: 1 2
Node 4 neighbours: 1 6 8 7
Node 5 neighbours: 1 6 7 8
Node 6 neighbours: 2 4 5
Node 7 neighbours: 4 5
Node 8 neighbours: 4 5
```

5. What is the change you will make in the `addedge` function so that Graph ADT could accept directed graphs. (Instead of accepting undirected graph, we need to accept directed graph). Write `addedge` altered function as your answer below.

```
void addedge(int u, int v, bool directed) {
    nodes[u].neighbours.push_back(v);
    if (!directed) {
        nodes[v].neighbours.push_back(u)
    }
}
```

Section 2 - Working out link prediction, no coding required

Let's assume graph in Fig.2 is a social network graph of a social media platform, where nodes denote people and edges between them indicate that they are connected as friends. **Node 1** and **Node 4** just became friends, which of the **neighbours of Node 1** will you **suggest for Node 4** (in other word predict which neighbour of Node 1 can have an edge with Node 4). Utilize the similarity function provided to justify the answer.

$$\text{Sim}(a, b) = \frac{\text{\# of shared neighbours between a, b}}{\text{Total neighbours in a, b}}$$

$$\text{Sim}(4, 2) = \frac{2}{5} = 0.4$$

$$\text{Sim}(4, 3) = \frac{1}{5} = 0.2$$

$$\text{Sim}(4, 5) = \frac{4}{4} = 1$$

Therefore, **Node 4** is predicted to have an edge with **Node 5**.