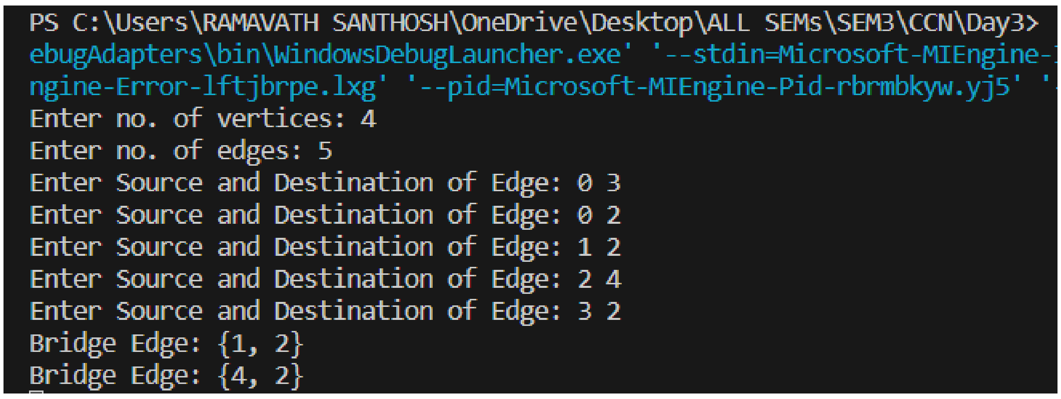
ASSIGNMENT 3

Question: Find all bridges present in the input graph.

CODE:

```
#include <iostream>
#include <map>
#include <vector>
#include <list>
using namespace std;

// Function to find bridge edges in the graph using DFS
void findBridge(map<int, list<int>> &adj, vector<bool> &visited, vector<int> &dis,
vector<int> &low, int i, vector<pair<int, int>> &ans, int parent)
{
    static int time = 0;
    visited[i] = true;
    dis[i] = low[i] = ++time; // Set discovery and low time for the current vertex

    for (auto it : adj[i])
    {                      // Traverse through all adjacent vertices of i
        if (it == parent) // Skip the edge back to parent vertex
            continue;

        if (visited[it])
        {
            low[i] = min(low[i], dis[it]); // Update low time if the vertex is
already visited
        }
        else
        {
            parent = i;
            findBridge(adj, visited, dis, low, it, ans, parent); // Recursively
call findBridge for unvisited adjacent vertex
            low[i] = min(low[i], low[it]);                       // Update low
time using child vertex's low time

            if (low[it] > dis[i])                // If the low time of the
adjacent vertex is greater than discovery time of the current vertex, it's a
bridge
                ans.push_back(make_pair(it, i)); // Store the bridge edge
        }
    }
}

int main()
{
    int V;
    cout << "Enter no. of vertices: ";
    cin >> V;
    int E;
    cout << "Enter no. of edges: ";
    cin >> E;
    map<int, list<int>> adj;
    // Constructing Adjacency List
    for (int i = 0; i < E; i++)
    {
        int s, d;
        cout << "Enter Source and Destination of Edge: ";
        cin >> s >> d;
        adj[s].push_back(d); // Add directed edges to the adjacency list
        adj[d].push_back(s); // Since the graph is undirected, add the reverse
edge as well
    }

    vector<bool> visited(V);
    vector<int> disc(V, -1);     // Discovery time of vertices
    vector<int> low(V, -1);      // The earliest visited vertex reachable from a
given vertex
    vector<pair<int, int>> edge; // Store bridge edges
    int parent = -1;

    // Call DFS Function
    for (int i = 0; i < V; i++)
    {
        if (!visited[i])
        {
            findBridge(adj, visited, disc, low, i, edge, parent); // Call
findBridge for unvisited vertices
        }
    }

    // Print the bridge edges
    for (auto it : edge)
        cout << "Bridge Edge: {" << it.first << ", " << it.second << "}" << endl;

    return 0;
}
```



```
PS C:\Users\RAMAVATH SANTHOSH\OneDrive\Desktop\ALL SEMs\SEM3\CCN\Day3>
ebugAdapters\bin\WindowsDebugLauncher.exe' '--stdin=Microsoft-MIEngine-I
ngine-Error-lftjbrpe.lxg' '--pid=Microsoft-MIEngine-Pid-rbrmbkyw.yj5' '
Enter no. of vertices: 4
Enter no. of edges: 5
Enter Source and Destination of Edge: 0 3
Enter Source and Destination of Edge: 0 2
Enter Source and Destination of Edge: 1 2
Enter Source and Destination of Edge: 2 4
Enter Source and Destination of Edge: 3 2
Bridge Edge: {1, 2}
Bridge Edge: {4, 2}
```