**Pratical-1**

**Implement the DAG in solidity**

**Objective:** A **Directed Acyclic Graph (DAG)** is a **finite graph** with **directed edges** and **no cycles**. The key objectives of using a DAG in computer science, mathematics, and real-world applications include:

**Code:**

/ SPDX-License-Identifier: MIT

pragma solidity ^0.8.0;

contract ShortDAG {

    mapping(uint => uint[]) public edges;

    mapping(uint => bool) public exists;

    event Node(uint id);

    event Link(uint from, uint to);

    function addNode(uint id) public {

        require(!exists[id], "Already exists");

        exists[id] = true;

        emit Node(id);

    }

    function addEdge(uint from, uint to) public {

        require(exists[from] && exists[to], "Nodes must exist");

        require(from != to, "No self-link");

        require(!hasPath(to, from), "Cycle detected");

        edges[from].push(to);

        emit Link(from, to);

    }

    function getLinks(uint id) public view returns (uint[] memory) {

        return edges[id];

    }

    function hasPath(uint from, uint to) internal view returns (bool) {

        if (from == to) ret

        urn true;

        for (uint i = 0; i < edges[from].length; i++) {

            if (hasPath(edges[from][i], to)) return true;

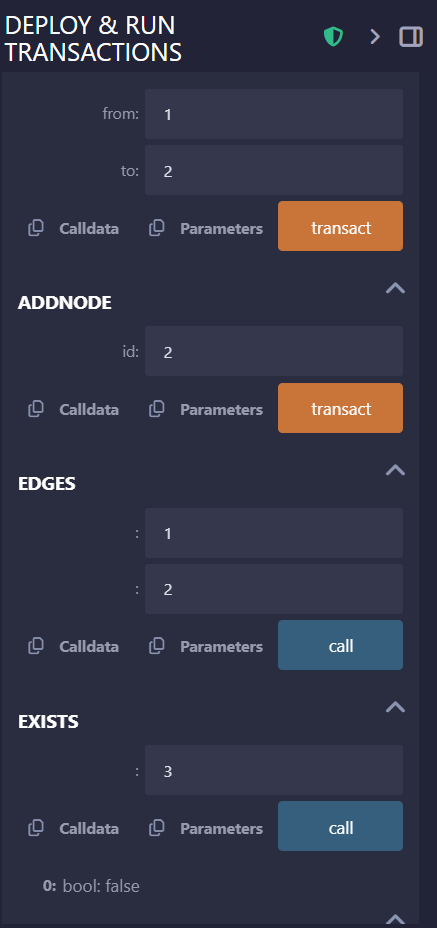
        }

        return false;

    }

}

**Output:**

****