**Blockchain & Cryptocurrency Lab**

**Pool Manager Lab**

**Salman Ahmed Khan 19K-1043**

**Solidity Code**

pragma solidity ^0.8.0;

contract PoolManager {

struct Transaction {

uint nonce;

string message;

}

Transaction public transaction;

uint public totalReward;

address poolManagerOwner;

uint poolManagerWallet;

mapping (address => bool) connectedNodes;

mapping (address => uint) rewards;

address[] rewardAddresses;

constructor() {

poolManagerOwner = msg.sender;

totalReward = 50;

transaction = Transaction(11, "HelloWorld");

}

modifier CheckOwner {

require (msg.sender == poolManagerOwner, "Only Pool Manager can perform this task");

\_;

}

modifier CheckNonceValueTransaction {

require (transaction.nonce > 0, "Can not mine transaction");

\_;

}

modifier IsTransactionMined {

require (transaction.nonce == 0, "Transaction already mined");

\_;

}

modifier IsNodeValid {

require (connectedNodes[msg.sender] == true, "Invalid Node Not Allowed");

\_;

}

function AddNode (address node) public CheckOwner {

connectedNodes[node] = true;

rewardAddresses.push(node);

}

function RemoveNode (address node) public CheckOwner {

connectedNodes[node] = false;

}

function random() public view returns (uint256) {

uint256 randomNumber = uint256(keccak256(abi.encodePacked(

block.timestamp,

block.difficulty,

block.coinbase,

msg.sender

)));

return randomNumber % 13;

}

function MineTransaction () public IsNodeValid CheckNonceValueTransaction {

uint256 \_nonce = random();

if (transaction.nonce == \_nonce) {

transaction.nonce = 0;

}

rewards[msg.sender] += 1;

}

function DistributeReward () public CheckOwner IsTransactionMined{

for (uint node = 0; node < rewardAddresses.length; node++) {

address \_node = rewardAddresses[node];

if (!connectedNodes[\_node]) continue;

// 5 % reward value is distributed

if (rewards[\_node] > 0 && rewards[\_node] < 5 && totalReward - 5 >= 0) {

rewards[\_node] = 5\*100;

totalReward -= 5;

}

// 20 % reward value is distributed

else if (rewards[\_node] > 5 && rewards[\_node] < 15 && totalReward - 10 >= 0) {

rewards[\_node] = 10\*100;

totalReward -= 10;

}

}

if (totalReward > 0)

poolManagerWallet = totalReward;

totalReward = 0;

}

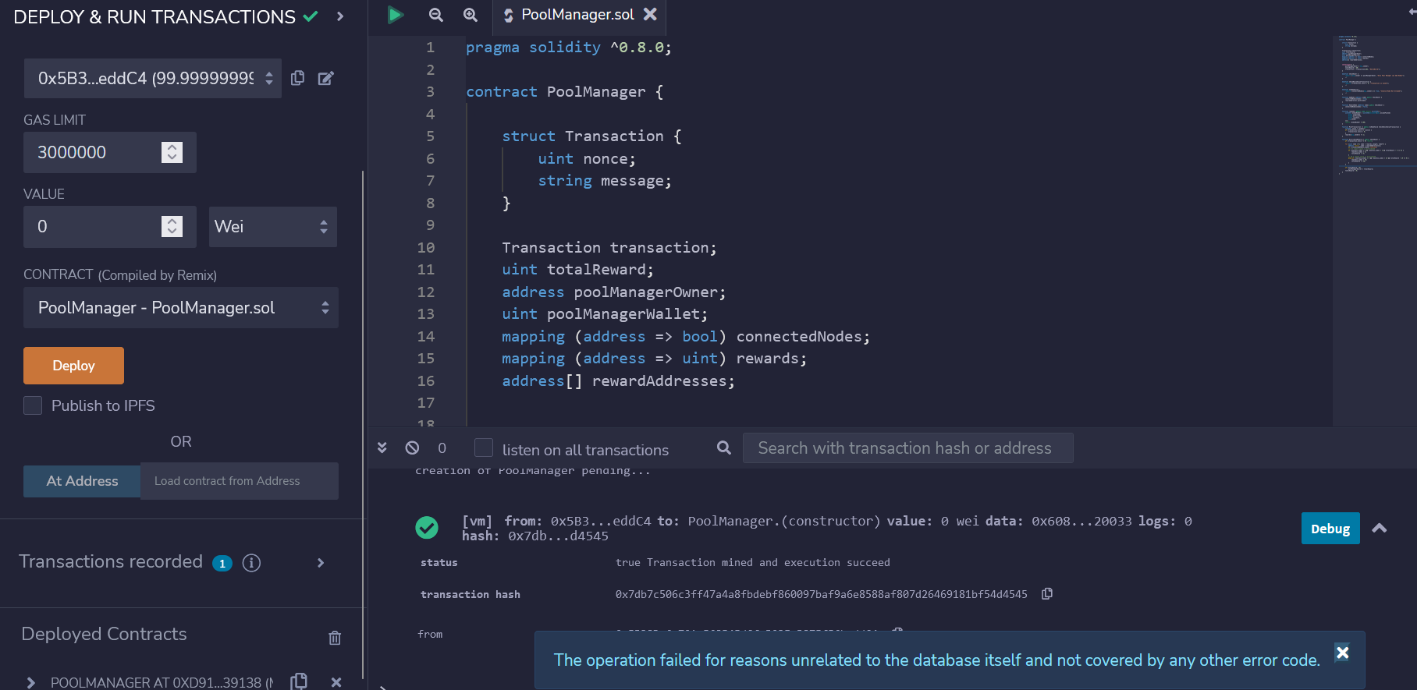
function getReward () IsNodeValid public view returns (uint){

return rewards[msg.sender];

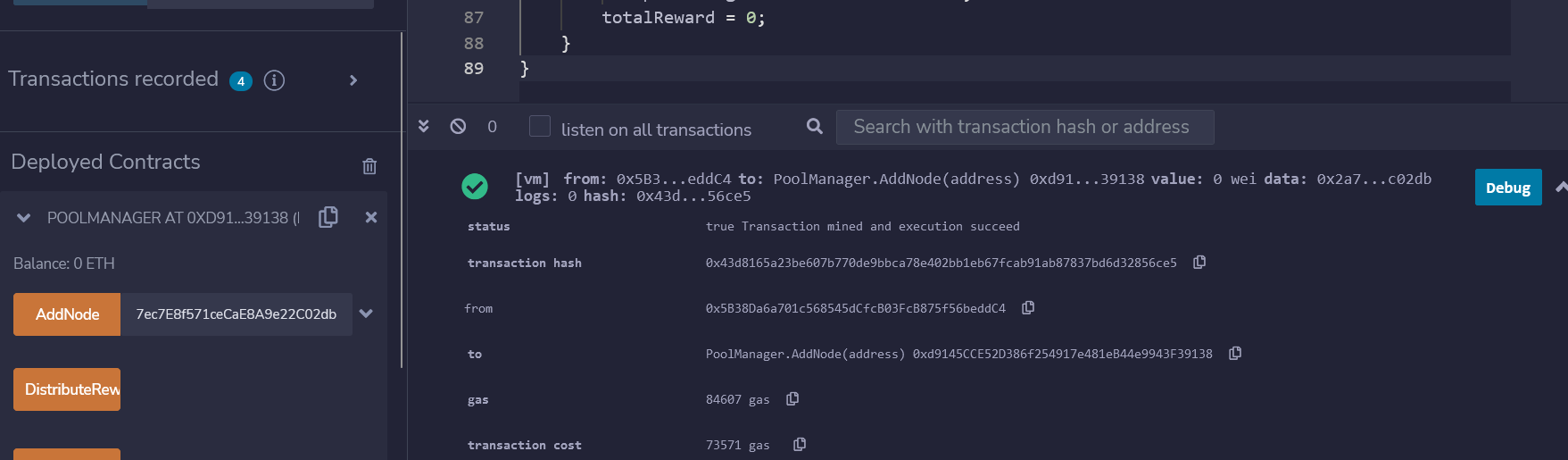
}

}

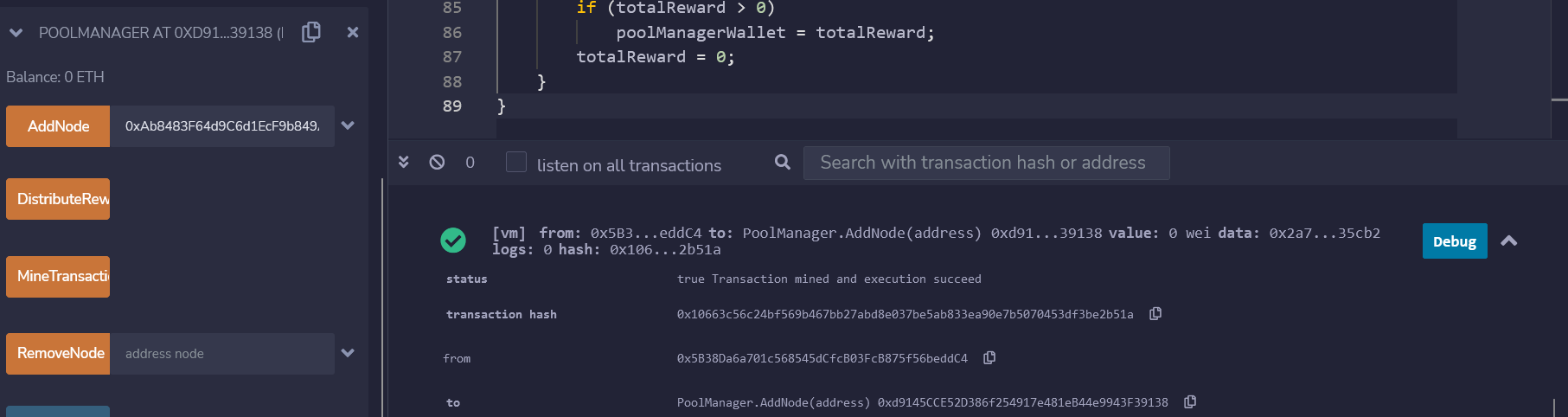
**Deploying Contract**



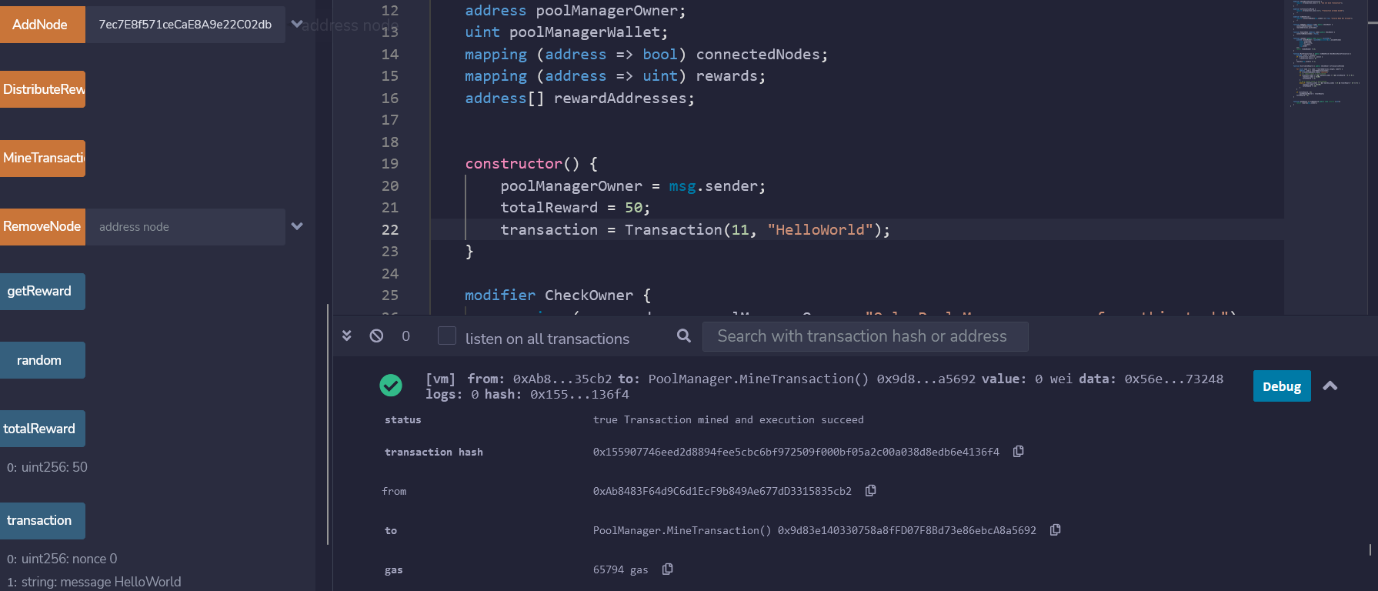
**Adding Node ‘A’**



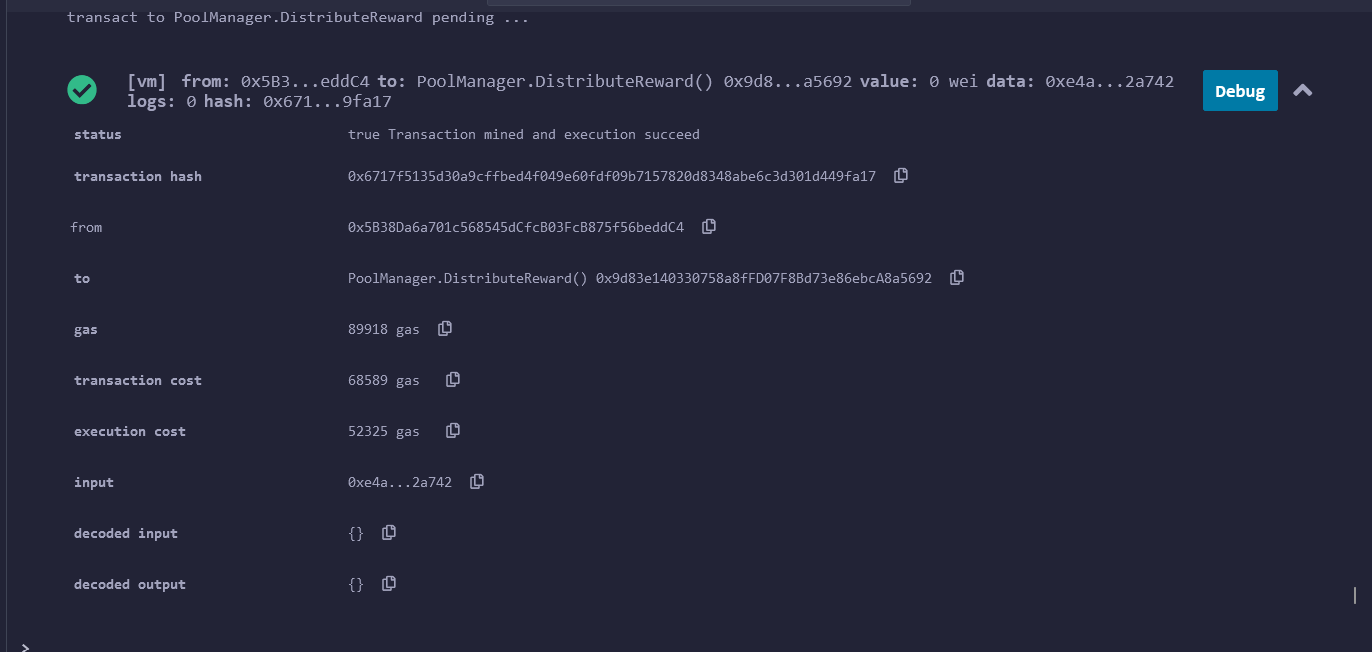
**Adding Node ‘B’**



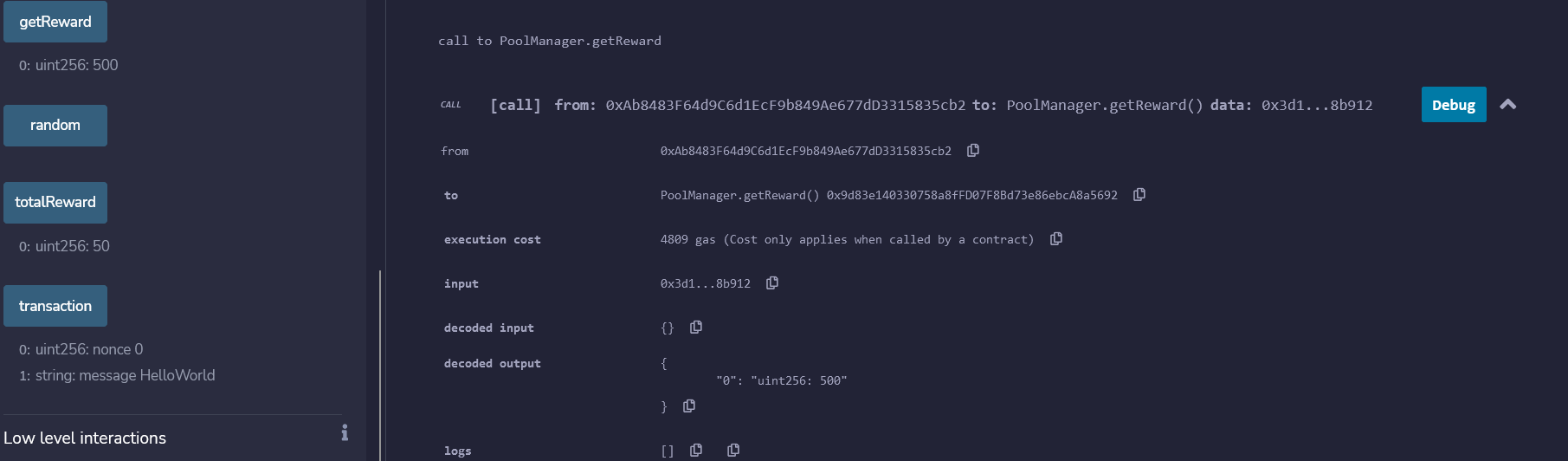
**Node A found the nonce:- The Transaction is mined**



**Pool Manager Distributed Reward**



**Node ‘A’ received Reward**

****