/\*\*

   \* @title ContractName

   \* @dev ContractDescription

   \* @custom:dev-run-script file\_path

   \* @custom:dev-run-script NatSpec tag

   \*/

// SPDX-License-Identifier: MIT

pragma solidity 0.8.18;

contract ProofOfWork {

    struct Block {

        uint256 previousBlockHash;

        bytes32 blockData;

        uint256 nonce;

    }

    uint256 public difficulty;

    uint256 public reward;

    bytes32 public target;

    address public miner;

    Block[] public blockchain;

    event NewProofOfWork(uint256 indexed nonce, bytes32 indexed blockHash, address indexed miner);

    constructor(uint256 \_difficulty, uint256 \_reward) {

        difficulty = \_difficulty;

        reward = \_reward;

        target = bytes32(2\*\*(256 - difficulty) - 1);

        miner = msg.sender;

    }

    function findProofOfWork(bytes32 \_blockData, uint256 \_previousBlockHash, uint256 \_nonce) public returns (bool) {

        bytes32 blockHash = calculateBlockHash(\_blockData, \_previousBlockHash, \_nonce);

        emit NewProofOfWork(\_nonce, blockHash, msg.sender);

        if (blockHash < target) {

            Block memory newBlock = Block(\_previousBlockHash, \_blockData, \_nonce);

            blockchain.push(newBlock);

            payable(miner).transfer(reward);

            return true;

        }

        return false;

    }

    function calculateBlockHash(bytes32 \_blockData, uint256 \_previousBlockHash, uint256 \_nonce) public pure returns (bytes32) {

        return sha256(abi.encodePacked(\_blockData, \_previousBlockHash, \_nonce));

    }

    function getLatestBlock() public view returns (uint256, bytes32, uint256) {

        require(blockchain.length > 0, "No blocks found");

        Block memory latestBlock = blockchain[blockchain.length - 1];

        return (latestBlock.previousBlockHash, latestBlock.blockData, latestBlock.nonce);

    }

}