BLOCKCHAIN DEVELOPER TEST

CHALLENGE 1: ETHEREUM SMART CONTRACT

Problem: Develop a simple 'Voting' smart contract on the Ethereum blockchain using Solidity. This contract should allow users to propose new items for voting, vote on them, and declare a winner. The smart contract should include the following functions:

1. **'proposeItem(string memory _item)':** This function should allow any user to propose a new item for voting.
2. **'voteForItem(uint256 _itemId)':** This function should allow any user to vote for a specific item using its ID. A user can only vote once.
3. **'getWinner()':** This function should return the ID and the name of the item with the most votes.

Submission: Please submit a zip file containing the .sol file for the Voting smart Contract.

Estimated time: 40 minutes

Sample Inputs and Outputs:

1. **'proposeItem("Blockchain")'**
    a. Output: A new item named 'Blockchain' is proposed for voting, the ID assigned to this item is 1.
2. **'voteForItem(1)'**
    a. Output: A vote is cast for the item with ID 1.
3. **'getWinner()'**
    a. Output: If 'Blockchain' has the highest votes, it returns **'{'ID': 1, 'Name': 'Blockchain'}'.**

Test Case:

1. Propose a few items for voting.
2. Cast votes for the items.
3. Check if the item with the highest vote is returned by the **'getWinner()'** function.

CHALLENGE 2: POLYGON BLOCKCHAIN TRANSACTION OPTIMIZATION

Problem: Write a Solidity function **'transferBatch(address[] calldata _to, uint256[] calldata _amounts)'** in a new smart contract that can perform multiple transfers in a single transaction on the Polygon blockchain to optimize gas cost. The function should perform a safety check that _to and _amounts arrays have the same length.

Submission: Please submit a zip file containing the .sol file with the optimized batch transfer function.

Estimated time: 40 minutes

Sample Inputs and Outputs:

1. **'transferBatch(["0xAb5801a7D398351b8bE11C439e05C5B3259aeC9B", "0x4E83362442B8d1bec281594CEA3050c8EB01311C"], [100, 200])'**
    a. Output: 100 tokens transferred to address "0xAb5801a7D398351b8bE11C439e05C5B3259aeC9B" and 200 tokens transferred to address "0x4E83362442B8d1bec281594CEA3050c8EB01311C".

Test Case:

1. Execute the 'transferBatch()' function with valid addresses and amounts.
2. Verify if the correct amount of tokens has been transferred to each address.

CHALLENGE 3: NODEJS AND BINANCE SMART CHAIN (BSC) INTEGRATION

Problem: Create a simple NodeJS script that integrates with a BSC Testnet smart contract using web3.js library. The smart contract's address and ABI will be given to you. Your script should:

1. Query the current total supply of the contract's token.
2. Call a function on the contract that mints a specified amount of tokens to a specific BSC address.

For the purpose of this test, use the following BSC Testnet BEP20 token contract address: **'0x0000000000000000000000000000000000000000'**

Assume the ABI has a function **'totalSupply()'** for querying the total supply and **'mint(address _to, uint256 _amount)'** for minting new tokens.

Submission: Please submit a zip file containing your NodeJS script.

Estimated time: 40 minutes

Sample Inputs and Outputs:

1. **'getTotalSupply()'**
    a. Output: Returns the current total supply of the token.
2. **'mint("0xAb5801a7D398351b8bE11C439e05C5B3259aeC9B", 1000)'**
    a. Output: Mints 1000 new tokens to the address "0xAb5801a7D398351b8bE11C439e05C5B3259aeC9B".

Test Case:

1. Run the **'getTotalSupply()'** function to get the current total supply.
2. Execute the **'mint()'** function with a valid address and amount.
3. Run the **'getTotalSupply()'** function again to verify if the total supply has increased by the amount specified in the **'mint()'** function.

SUBMISSION GUIDELINES:

Submit your code as a ZIP file. Ensure the zip file includes a README with instructions on how to run your application, how you have used each library in the task, and their purpose.

EVALUATION CRITERIA:

- Adherence to coding standards.
- The efficiency of the solution.
- Correctness of the implementation based on the sample input and expected output.
- Proper handling of edge cases.
- Clarity of the AWS deployment guide.

Note: Don’t worry about copyright. If you are not selected, this task is yours to showcase on your GitHub forever.