**Birla Institute of Technology & Science, Pilani**

**Work-Integrated Learning Programmes Division**

**First Semester 2022-2023**

**COMPREHENSIVE EXAMINATION**

**(EC-3 Regular)**

Course No. : SE ZG569

Course Title : BLOCK CHAIN TECHNOLOGIES AND SYSTEMS

Nature of Exam : Open Book

No. of Pages = 2

# No. of Questions = 7

Weightage : 40%

Duration : 2½ Hours

Date of Exam : Sunday, 27/12/2022 (AN)

Note:

1. Please follow all the *Instructions to Candidates* given on the cover page of the answer book.
2. All parts of a question should be answered consecutively. Each answer should start from a fresh page.
3. Assumptions made if any, should be stated clearly at the beginning of your answer.
4. Answer the following
5. Discuss why the developers of Ethereum created a new language called solidity to write smart contracts instead of taking any other higher-level language such as GO. [2]
6. The malicious code in Bitcoin can affect the peer node on which it is running. Due to this, Bitcoin has allowed limited operations in Smart contracts. How does Ethereum prevent the malicious code in the smart contract to not affect the peer machines? [2]
7. Answer the following
   1. Discuss in detail the purpose and the working (with example) of all the fields of Ethereum transaction. [2]
   2. Why there is no sender field in ethereum transaction? [1]
   3. How do we keep track of nonce. Discuss the issues of using getTransactionCount() to get the current nonce value.

[1]

1. Consider the DNS database given in Table 1. The domain name system (DNS) converts human readable URLs (google.com) into IP addresses like (172.18.39.72). Storing this key-value (IP address-domain name) mapping in a linear data structure would require long search time and also it is not temper proof from malevolent modifications. To overcome these limitations, we decide to store the database using the Merkle Particia tree used by the Ethereum Blockchain. Construct an optimal MPT for the given database. Show your MPT using a neat diagram containing the node details and their relationships. Note that instead of prefixes, here we have common suffixes. For sake of simplicity you can consider the byte based representation instead of nibble based representation for each character. [4]

|  |  |
| --- | --- |
| **URL (Key)** | **IP address (Value)** |
| google.com | 142.251.107.17 |
| mail.google.com | 172.217.204.83 |
| maps.google.com | 108.177.12.138 |
| apple.com | 17.253.144.10 |
| amazon.com | 17.253.124.9 |
| bits-pilani.ac.in | 14.139.243.20 |
| iit.ac.in | [218.248.46.85](https://whatismyipaddress.com/ip/218.248.46.85) |
| drdo.gov.in | 25.253.144.20 |

1. Answer the following
2. What do we mean by private variables in smart contract? Can we use such variable to protect the data from unauthorized access?
3. Write the steps and requirements for deploying a smart contract on the Ethereum blockchain network?
4. Write down at least four use cases for logging the events in the smart contracts. [6]
5. Fill in the solidity functions for deposit(), withdraw(), and transfer() according the descriptions provided [6]

pragma solidity ^0.8.0;

contract SimpleBank {

mapping (address => uint) private balances;

address public owner;

event LogDepositMade(address accountAddress, uint amount);

constructor () {

owner = msg.sender;

}

function **deposit()** public payable returns (uint) {

**// increase balance of message sender, emit LogDepositMade event and return the new balance**

**// Write your code below!**

}

function withdraw(uint withdrawAmount) public payable returns (uint remainingBal) {

**// Reduce the balance of the user by withdrawAmount (are any checks needed here?), transfer the amount specified by withdrawAmount to the user, and return the new balance**

**// Write your code below!**

}

function transfer(address recipient, uint transferAmount) public returns (uint remainingBal) {

**// Subtract sender balance, and add it to receiver balance (Any checks required here?). Return the new balance of the sender**

**// Write your code below!**

}

function balance() view public returns (uint) {

        return balances[msg.sender];

    }

}

1. You have taken a blockchain course and therefore, you should be able to generate your own cryptocurrency using solidity. Let's assume that the initial coin offering (ICO) is equivalent to ). You should be able to transfer some coins (not more than ICO) to any other address based on the requirements. You can allow someone else to transfer the coins on your behalf if required. In this situation, you have to approve someone to do that. We are providing you with the partial code to implement the functionality discussed above.



Complete the smart contract by writing the code for the following questions. No need to write the complete source code. Provide the code only for question I to IV.

1. maintain the information on who has how many coins.
2. In case, you are approving someone to transfer the coins on your behalf, maintain the number of coins you are allowing that person to do that. Your address should be associated with the approved person's address.
3. Write the functionality of approving someone to send the coins on your behalf. Make sure you are storing the information in a log. Also, you have to make sure that the address you are approving is a valid address.
4. Write the functionality of transferring coins from an address to another address through the approved person. Make sure that the address you are sending to is a valid address.

pragma solidity ^0.5.0;

contract Token {

string public name = "DApp Token";    // Cyptocurrency Name

string public symbol = "DAPP"; // Cryptocurrency Symbol

uint256 public decimals = 18;

uint256 public totalSupply;

**Write Answer for I**

**Write Answer for II**

event Transfer(address indexed \_from, address indexed \_to, uint256 \_value);

event Approval(address indexed \_owner, address indexed \_spender, uint256 \_value);

constructor() public {

totalSupply = 1000000 \* (10 \*\* decimals);

balanceOf[msg.sender] = totalSupply;

}

function transfer(address \_to, uint256 \_value) public returns (bool success) {

require(balanceOf[msg.sender] >= \_value);

\_transfer(msg.sender, \_to, \_value);

return true;

}

function \_transfer(address \_from, address \_to, uint256 \_value) internal {

require(\_to != address(0));

balanceOf[\_from] -= \_value;

balanceOf[\_to] += \_value;

emit Transfer(\_from, \_to, \_value);

}

function approve(address \_spender, uint256 \_value) public returns (bool success){

**Write an answer for III**

}

function transferFrom(address \_from, address \_to, uint256 \_value) public returns (bool success){

**Write an answer for IV**

}

**[8]**

1. Mr. X is an old person in BITS Pilani campus who has been driving a manual rickshaw for many years. However, due to old age, it is becoming difficult for him to earn sufficient money for his livelihood. After seeing his condition, you decided to raise funds for him by writing and deploying a smart contract over the Ethereum blockchain so that he can start a small business selling fruits instead of driving a manual rickshaw. Each donor is allowed to donate more than once. Each donor's information (donor address, amount, and date) should be maintained in the contract except for the one who wants to donate the fund anonymously. While receiving a donation, maintain an ethereum log (about the donor and amount) as well. Each donor should be able to see the details of the donations made by him/her till the present time. Write a smart contract to receive funds (donations) in form of ether from others and transfer the collected ether to Mr. X's account (say X) for the following contract conditions. Assume that depositors have already verified the wallet address of Mr. X.

A) The amount should be transferred to Mr. X only when the MAX\_AMOUNT number of ethers is deposited to the smart contract. Only Mr. X should be able to extract the ethers from the smart contract. If more than MAX\_AMOUNT is deposited the smart contract will hold it for future use. Also, maintain a log while withdrawing the ether amount.

B) The whole amount deposited in the smart contract should be transferred to Mr. X immediately after MAX\_DAYS from the deployment of the smart contract. Mr. X should only be able to fetch ether from the smart contract to his account if and only if the owner of the smart contract has approved it. While approving the address, make sure that Mr. X’s address is a valid address. [8]

\*\*\*\*\*\*\*