**Exercises for Exam Simulation- May 19th**

**QUESTIONS**

**Question. Gi**ve a definition of decentralised ledger. Give an example of a smart contract that contains one or more functions that modify the ledger. ( max 10 lines-100 words)

If I change balance or create a new smart contract or create a wallet I am changing the ledger, but if I simply check info I am not changing the ledger. Make simple examples -> if I send token from one to another wallet

**Question. L**ist and explain three (3) properties of the distributed ledger, making an example of application for each of them. (max 15 lines or 150 words).

Immutability-> example PoW, to modify something u need the majority consensus of the node OR double spending, distributed (in the meaning of before), transparency-> example, I can use for a notary, decentralized

**Question. E**xplain what are Elliptic Curves and how they are used in Bitcoin. (Max 10 lines 100 words)

Elliptic curves are mathematical structures defined by equations like *y² = x³ + ax + b*. Bitcoin uses the curve **secp256k1** for its public-key cryptography. A private key is a random number, and the corresponding public key is a point on the curve. This provides secure, asymmetric encryption for signing and verifying transactions.

**Question.** Explain the following concepts: smart contract, Dapp, and wallet. Then describe a use case that uses at least two of them.

A **smart contract** is self-executing code on a blockchain. A **DApp** (Decentralized App) uses smart contracts as its backend. A **wallet** stores private keys to interact with the blockchain.  
**Use Case**: A DApp for NFT marketplaces uses smart contracts to manage ownership, while users access it via wallets like MetaMask to sign transactions and make purchases.

A wallet is a particular Dapp, but not all dapp are wallet

**Exercises**

1. **EXERCISE. Create a ERC 20 token with the following features:**

* 10,000 circulation
* Name Symbol: TKX
* Functions: the wallet should be able to transfer the tokens and to get the balance

After it, explain your choice and if you are using modifiers or constructors and suggest another possible approach to creation respect to your choice.

1. **Exercise n. 6 DIGITAL VENDING MACHINES**

Run the code explain by example what the functions refill and purchase are doing

[vm] from: 0x5B3... → to: VendingMachine.(constructor) 0x123...abc

That 0x123...abc is the contract’s address you need to check cupcake balance.

If you're buying 2 cupcakes, you must **send at least 2 ETH** with your transaction.

* In the VALUE field:
  + Enter: 2
  + Next to it, make sure the dropdown is set to **ether**

✅ Now Remix will send **2 Ether** when you call the function.

* ow, click the button next to:

solidity

CopiaModifica

purchase(uint amount)

* + Input 2
  + Click it

🎉 The transaction should now succeed.

If u look at balance is 98!!!

**pragma solidity 0.8.7;**

**contract VendingMachine {**

**// Declare state variables of the contract**

**address public owner;**

**mapping (address => uint) public cupcakeBalances;**

**// When 'VendingMachine' contract is deployed:**

**// 1. set the deploying address as the owner of the contract**

**// 2. set the deployed smart contract's cupcake balance to 100**

**constructor() {**

**owner = msg.sender;**

**cupcakeBalances[address(this)] = 100;**

**}**

**// Allow the owner to increase the smart contract's cupcake balance**

**function refill(uint amount) public {**

**require(msg.sender == owner, "Only the owner can refill.");**

**cupcakeBalances[address(this)] += amount;**

**}**

**// Allow anyone to purchase cupcakes**

**function purchase(uint amount) public payable {**

**require(msg.value >= amount \* 1 ether, "You must pay at least 1 ETH per cupcake");**

**require(cupcakeBalances[address(this)] >= amount, "Not enough cupcakes in stock to complete this purchase");**

**cupcakeBalances[address(this)] -= amount;**

**cupcakeBalances[msg.sender] += amount;**

**}**

**}**