

# Subject Name Solutions

4361603 – Summer 2024

Semester 1 Study Material

*Detailed Solutions and Explanations*

## Question 1(a) [3 marks]

Explain benefits of using distributed ledger systems.

### Solution

Table 1: Benefits of Distributed Ledger Systems

Benefit	Description
<b>Transparency</b>	All participants can view transaction history
<b>Security</b>	Cryptographic protection against tampering
<b>Decentralization</b>	No single point of failure or control
<b>Immutability</b>	Records cannot be altered once confirmed

### Mnemonic

“T-S-D-I” (Transparent, Secure, Decentralized, Immutable)

## Question 1(b) [4 marks]

Define: 1) Blockchain 2) Distributed systems

### Solution

Table 2: Key Definitions

Term	Definition
<b>Blockchain</b>	A chain of blocks containing transaction data, linked using cryptographic hashes
<b>Distributed Systems</b>	Network of independent computers working together as a single system

#### Key Features:

- **Blockchain:** Uses hash pointers, consensus mechanisms, and merkle trees
- **Distributed Systems:** Fault tolerance, scalability, and resource sharing

### Mnemonic

“Chain-Hash-Consensus” for Blockchain, “Network-Independent-Together” for Distributed

## Question 1(c) [7 marks]

Illustrate CAP theorem with the help of Blockchain network.

### Solution

Table 3: CAP Theorem Components

Property	Description	Blockchain Context
<b>Consistency</b>	All nodes see same data	All nodes have identical ledger
<b>Availability</b>	System remains operational	Network stays accessible
<b>Partition Tolerance</b>	Works despite network failures	Continues during node disconnections

Diagram:

### Mermaid Diagram (Code)

```
{Shaded}
{Highlighting}[]
graph TD
    A[CAP Theorem] --{-}{-}{ B[Consistency]}
    A --{-}{-}{ C[Availability] }
    A --{-}{-}{ D[Partition Tolerance]}
    B --{-}{-}{ E[Bitcoin prioritizes C+P]}
    C --{-}{-}{ F[Some systems choose A+P]}
    D --{-}{-}{ G[Essential for blockchain]}
{Highlighting}
{Shaded}
```

Key Points:

- **Trade-off:** Can only achieve 2 out of 3 properties simultaneously
- **Blockchain Choice:** Most blockchains choose Consistency + Partition Tolerance
- **Example:** Bitcoin may become temporarily unavailable but maintains consistency

### Mnemonic

“CAP-2-out-of-3” (Choose Any 2 Properties out of 3)

## Question 1(c) OR [7 marks]

List and explain applications of blockchain network.

### Solution

Table 4: Blockchain Applications

Application	Description	Example
<b>Cryptocurrency</b>	Digital money transactions	Bitcoin, Ethereum
<b>Supply Chain</b>	Track products from origin	Walmart food tracing
<b>Healthcare</b>	Secure patient records	Medical data sharing
<b>Voting</b>	Transparent elections	Estonia e-voting
<b>Real Estate</b>	Property ownership records	Land registries

Key Benefits:

- **Transparency:** All transactions visible to participants
- **Security:** Cryptographic protection against fraud
- **Efficiency:** Reduced intermediaries and costs

### Mnemonic

“C-S-H-V-R” (Crypto, Supply, Health, Vote, Real estate)

## Question 2(a) [3 marks]

Define and explain a permissionless blockchain in detail.

### Solution

**Definition:** A blockchain where anyone can participate without requiring permission from a central authority.

Table 5: Permissionless Blockchain Features

Feature	Description
<b>Open Access</b>	Anyone can join and participate

<b>Public Verification</b>	All transactions are publicly verifiable
<b>Decentralized</b>	No central controlling authority

**Key Characteristics:**

- **Consensus:** Uses proof-of-work or proof-of-stake
- **Examples:** Bitcoin, Ethereum mainnet

**Mnemonic**

“Open-Public-Decentralized” (OPD)

Question 2(b) [4 marks]

Draw a figure and provide a brief explanation of a data structure of a blockchain.

**Solution**

**Diagram: Blockchain Data Structure**

```
+{-}{-}{-}{-}{-}{-}{-}{-}{-}{-}{-}{-}{-}{-}{-}{-}{-}{-}{-}{-}{-}{-}{-}+    +{-}{-}{-}{-}{-}{-}{-}{-}{-}{-}{-}{-}{-}{-}{-}{-}{-}{-}{-}{-}{-}{-}{-}+  
|      Block 1          |      |      Block 2          |      |      Block 3          |  
|{-}{-}{-}{-}{-}{-}{-}{-}{-}{-}{-}{-}{-}{-}{-}{-}{-}{-}{-}{-}{-}{-}{-}|    |{-}{-}{-}{-}{-}{-}{-}{-}{-}{-}{-}{-}{-}{-}{-}{-}{-}{-}{-}{-}{-}{-}{-}  
| Previous Hash: 0     |{-}{-}{-}| Previous Hash: H1   |{-}{-}{-}| Previous Hash: H2   |}  
| Merkle Root: MR1     |      | Merkle Root: MR2     |      | Merkle Root: MR3     |  
| Timestamp: T1        |      | Timestamp: T2        |      | Timestamp: T3        |  
| Nonce: N1            |      | Nonce: N2            |      | Nonce: N3            |  
| Transactions: TX1    |      | Transactions: TX2    |      | Transactions: TX3    |  
+{-}{-}{-}{-}{-}{-}{-}{-}{-}{-}{-}{-}{-}{-}{-}{-}{-}{-}{-}{-}{-}{-}{-}+    +{-}{-}{-}{-}{-}{-}{-}{-}{-}{-}{-}{-}{-}{-}{-}{-}{-}{-}{-}{-}{-}{-}{-}
```

The diagram illustrates the structure of three consecutive blocks in a blockchain. Each block is represented as a container with several key components:

- Previous Hash:** Links the current block to the previous one (e.g., 0, H1, H2).
- Merkle Root:** A summary hash of all transactions in the block (MR1, MR2, MR3).
- Timestamp:** The time when the block was created (T1, T2, T3).
- Nonce:** A number used once for proof-of-work (N1, N2, N3).
- Transactions:** The actual data being recorded (TX1, TX2, TX3).

The blocks are connected by their hashes, forming a chain.

**Mnemonic**

“P-M-T-N” (Previous, Merkle, Time, Nonce)

Question 2(c) [7 marks]

Explain the core components of blockchain with suitable diagrams.

Solution		
Table 6: Core Components of Blockchain		
Component	Function	Purpose
Blocks	Data containers	Store transaction information
Hash Functions	Create digital fingerprints	Ensure data integrity
Merkle Trees	Transaction summaries	Efficient verification
Consensus Mechanism	Agreement protocol	Validate new blocks
Digital Signatures	Identity verification	Authenticate transactions

### Diagram: Merkle Tree Structure

#### Mermaid Diagram (Code)

```
{Shaded}
{Highlighting}[]
graph TD
    A[Root Hash] --{-}{-}{-} B[Hash AB]}
    A --{-}{-}{-} C[Hash CD]}
    B --{-}{-}{-} D[Hash A]}
    B --{-}{-}{-} E[Hash B]}
    C --{-}{-}{-} F[Hash C]}
    C --{-}{-}{-} G[Hash D]}
    D --{-}{-}{-} H[Transaction A]}
    E --{-}{-}{-} I[Transaction B]}
    F --{-}{-}{-} J[Transaction C]}
    G --{-}{-}{-} K[Transaction D]}
{Highlighting}
{Shaded}
```

#### Key Points:

- **Immutability:** Hash functions make tampering detectable
- **Efficiency:** Merkle trees allow fast verification
- **Decentralization:** Consensus mechanisms eliminate central authority

#### Mnemonic

“B-H-M-C-D” (Blocks, Hash, Merkle, Consensus, Digital)

### Question 2(a) OR [3 marks]

Define and explain permissioned blockchain in detail.

#### Solution

**Definition:** A blockchain where participation requires explicit permission from a governing authority.

Table 7: Permissioned Blockchain Features

Feature	Description
<b>Restricted Access</b>	Only authorized users can participate
<b>Private Network</b>	Controlled membership
<b>Centralized Control</b>	Governing body manages permissions

#### Key Characteristics:

- **Privacy:** Enhanced confidentiality for sensitive data
- **Performance:** Faster transactions due to fewer validators
- **Examples:** Hyperledger Fabric, R3 Corda

#### Mnemonic

“Restricted-Private-Centralized” (RPC)

### Question 2(b) OR [4 marks]

Explain types of wallets in the context of blockchain. Also discuss the factors to be considered while selecting wallet for the specific need.

## Solution

Table 8: Types of Blockchain Wallets

Wallet Type	Description	Security Level
<b>Hot Wallets</b>	Connected to internet	Medium
<b>Cold Wallets</b>	Offline storage	High
<b>Hardware Wallets</b>	Physical devices	Very High
<b>Paper Wallets</b>	Printed keys	High (if stored safely)

### Selection Factors:

- **Security Requirements:** Higher value needs better security
- **Frequency of Use:** Regular use favors hot wallets
- **Technical Expertise:** Simple wallets for beginners

## Mnemonic

“H-C-H-P” (Hot, Cold, Hardware, Paper)

## Question 2(c) OR [7 marks]

Explain sidechain in detail with suitable diagrams.

## Solution

**Definition:** A separate blockchain that is attached to a parent blockchain using a two-way peg.

**Diagram: Sidechain Architecture**

### Mermaid Diagram (Code)

```
{Shaded}
{Highlighting}[]
graph LR
    A[Main Chain] <--> B[Two-Way Peg]
    B <--> C[Sidechain]
    A <--> D[Bitcoin/Ethereum]
    C <--> E[Specialized Functions]
    E <--> F[Smart Contracts]
    E <--> G[Privacy Features]
    E <--> H[Faster Transactions]
{Highlighting}
{Shaded}
```

Table 9: Sidechain Benefits

Benefit	Description
<b>Scalability</b>	Reduces load on main chain
<b>Experimentation</b>	Test new features safely
<b>Specialized Functions</b>	Custom applications
<b>Interoperability</b>	Connect different blockchains

### Key Mechanisms:

- **Two-Way Peg:** Allows asset transfer between chains
- **SPV Proofs:** Simplified payment verification
- **Federated Control:** Multiple parties manage transfers

## Mnemonic

“S-E-S-I” (Scalability, Experimentation, Specialized, Interoperability)

Question 3(a) [3 marks]

With respect to transaction in a blockchain network, define the terms “Confirmation” and “Finality”.

Solution

Table 10: Transaction States

Term	Definition
Confirmation	Number of blocks built on top of transaction block
Finality	Point where transaction becomes irreversible

Key Points:

- **Confirmation Count:** More confirmations = higher security
- **Bitcoin Standard:** 6 confirmations for high-value transactions
- **Finality Types:** Probabilistic (Bitcoin) vs Absolute (some PoS systems)

Mnemonic

“Count-Blocks-Security” for Confirmation, “Irreversible-Point” for Finality

Question 3(b) [4 marks]

Differentiate Proof of Work and Proof of Stake.

Solution

Table 11: PoW vs PoS Comparison

Aspect	Proof of Work (PoW)	Proof of Stake (PoS)
Resource	Computational power	Stake ownership
Energy Use	High	Low
Security	Hash rate dependent	Stake dependent
Rewards	Mining rewards	Staking rewards
Examples	Bitcoin, Ethereum (old)	Ethereum 2.0, Cardano

Key Differences:

- **Mechanism:** PoW uses mining, PoS uses validators
- **Environmental Impact:** PoS is more eco-friendly
- **Barriers to Entry:** PoS requires initial stake, PoW needs hardware

Mnemonic

“Work-vs-Stake” (Computational Work vs Financial Stake)

Question 3(c) [7 marks]

With respect to blockchain network, explain 51% attack.

Solution

**Definition:** An attack where a single entity controls more than 50% of the network’s mining power or stake.

**Diagram: 51% Attack Scenario**

Mermaid Diagram (Code)

```
{Shaded}
{Highlighting}[]
graph LR
    A[Network Hash Rate {--}{ B[Honest Miners 49%\]]
    A {--}{ C[Attacker 51%\]]
```

C {-}{-}{ } D[Can Create Longer Chain]  
 D {-}{-}{ } E[Double Spending]  
 D {-}{-}{ } F[Transaction Reversal]  
 D {-}{-}{ } G[Block Withholding]  
 {Highlighting}  
 {Shaded}

Table 12: Attack Capabilities and Limitations

Can Do	Cannot Do
Double spend own coins	Steal others' coins
Reverse recent transactions	Create coins from nothing
Block specific transactions	Change consensus rules
Fork the blockchain	Access private keys

#### Prevention Measures:

- **Diversified Mining:** Encourage multiple mining pools
- **Checkpoint Systems:** Periodic finality markers
- **Economic Incentives:** Make attacks unprofitable

#### Impact:

- **Network Disruption:** Temporary service interruption
- **Economic Loss:** Reduced trust and value
- **Recovery:** Network usually recovers after attack ends

#### Mnemonic

“Majority-Control-Attack” (51% = Majority Control = Attack Power)

### Question 3(a) OR [3 marks]

Define the terms “Hard fork” and “Soft fork”

#### Solution

Table 13: Fork Types

Fork Type	Definition	Compatibility
<b>Hard Fork</b>	Non-backward compatible protocol change	Not compatible
<b>Soft Fork</b>	Backward compatible protocol change	Compatible

#### Key Characteristics:

- **Hard Fork:** Creates new blockchain branch, requires all nodes to upgrade
- **Soft Fork:** Tightens rules, old nodes can still operate

#### Examples:

- **Hard Fork:** Bitcoin Cash split from Bitcoin
- **Soft Fork:** SegWit activation in Bitcoin

#### Mnemonic

“Hard-Breaks-Compatibility” vs “Soft-Keeps-Compatibility”

### Question 3(b) OR [4 marks]

List various types of consensus mechanisms and explain any one in detail.

## Solution

Table 14: Consensus Mechanisms

Mechanism	Description	Energy Use
<b>Proof of Work</b>	Computational puzzle solving	High
<b>Proof of Stake</b>	Stake-based validation	Low
<b>Delegated PoS</b>	Voted representatives validate	Very Low
<b>Proof of Authority</b>	Pre-approved validators	Minimal

### Detailed Explanation - Proof of Stake (PoS):

#### Process:

- **Validator Selection:** Based on stake amount and randomization
- **Block Creation:** Selected validator proposes new block
- **Validation:** Other validators verify and attest to block
- **Rewards:** Validators earn fees and new tokens

**Advantages:** Lower energy consumption, reduced centralization risk **Disadvantages:** “Nothing at stake” problem, initial distribution issues

## Mnemonic

“Stake-Select-Validate-Reward” (PoS Process)

## Question 3(c) OR [7 marks]

With respect to blockchain network, explain sybil attack.

## Solution

**Definition:** An attack where a single adversary creates multiple fake identities to gain disproportionate influence in the network.

### Diagram: Sybil Attack Structure

#### Mermaid Diagram (Code)

```
{Shaded}
{Highlighting}[]
graph TD
    A[Attacker] --> B[Fake Identity 1]
    A --> C[Fake Identity 2]
    A --> D[Fake Identity 3]
    A --> E[Fake Identity N]
    B --> F[Network Influence]
    C --> F
    D --> F
    E --> F
    F --> G[Consensus Manipulation]
{Highlighting}
{Shaded}
```

Table 15: Attack Methods and Defenses

Attack Method	Description	Defense
<b>Identity Flooding</b>	Create many fake nodes	Proof of Work/Stake
<b>Routing Manipulation</b>	Control network paths	Reputation systems
<b>Consensus Disruption</b>	Influence voting	Resource requirements

**Impact on Blockchain:**

- **Network Partitioning:** Isolate honest nodes
- **Double Spending:** Facilitate fraudulent transactions
- **Consensus Failure:** Prevent network agreement

**Prevention Mechanisms:**

- **Resource Requirements:** PoW/PoS make attacks expensive
- **Identity Verification:** KYC/AML procedures
- **Network Monitoring:** Detect suspicious behavior patterns
- **Reputation Systems:** Track node behavior over time

**Real-world Examples:**

- **P2P Networks:** BitTorrent, Gnutella vulnerabilities
- **Social Networks:** Fake account creation
- **Blockchain:** Potential threat to permissionless networks

**Mnemonic**

“Single-Multiple-Influence” (Single Attacker, Multiple Identities, Network Influence)

**Question 4(a) [3 marks]**

Define the terms “Merkle Tree” and “Hyperledger”.

**Solution**

Table 16: Key Definitions

Term	Definition
<b>Merkle Tree</b>	Binary tree of hashes that efficiently summarizes all transactions
<b>Hyperledger</b>	Open-source blockchain platform hosted by Linux Foundation

**Key Features:**

- **Merkle Tree:** Enables efficient verification without downloading full blockchain
- **Hyperledger:** Enterprise-focused, modular architecture, multiple frameworks

**Mnemonic**

“Tree-Hash-Efficient” for Merkle, “Enterprise-Modular-Linux” for Hyperledger

**Question 4(b) [4 marks]**

Explain classic Byzantine generals problem in detail.

**Solution**

**Scenario:** Multiple generals must coordinate attack on a city, but some may be traitors.

Table 17: Problem Components

Component	Description
<b>Generals</b>	Network nodes/participants
<b>Messages</b>	Transactions/communications
<b>Traitors</b>	Malicious/faulty nodes
<b>Consensus</b>	Agreement on action

**Solution Requirements:**

- **Agreement:** All honest generals decide on same action
- **Validity:** If all honest generals want to attack, they should attack
- **Termination:** Decision must be reached in finite time

**Blockchain Relevance:** Ensures network agreement despite malicious nodes

### Mnemonic

“Generals-Messages-Traitors-Consensus” (GMTC)

### Question 4(c) [7 marks]

Explain the process of Merkle tree creation with suitable example and supporting diagrams.

#### Solution

##### Process Steps:

1. Hash each transaction individually
2. Pair hashes and hash the pairs
3. Continue until single root hash remains

##### Example: 4 Transactions

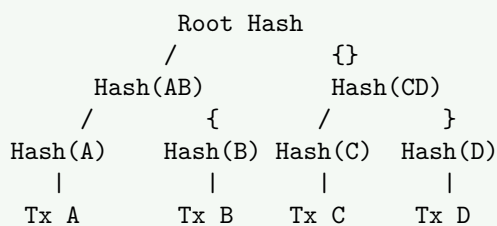


Table 18: Merkle Tree Benefits

Benefit	Description
<b>Efficiency</b>	Verify transactions without full data
<b>Security</b>	Any change affects root hash
<b>Scalability</b>	Log(n) verification complexity

##### Verification Process:

- To verify Tx A: Need Hash(B), Hash(CD), and Root Hash
- Path verification:  $\text{Hash(A)} + \text{Hash(B)} = \text{Hash(AB)}$
- $\text{Hash(AB)} + \text{Hash(CD)} = \text{Root Hash}$

##### Applications:

- **Bitcoin:** Block headers contain Merkle root
- **SPV Clients:** Light wallets use Merkle proofs
- **Git:** Version control system uses similar structure

### Mnemonic

“Hash-Pair-Repeat-Root” (Merkle Tree Creation Process)

### Question 4(a) OR [3 marks]

List various types of Hyperledger projects.

#### Solution

Table 19: Hyperledger Projects

Project	Type	Purpose
<b>Fabric</b>	Framework	Permissioned blockchain platform
<b>Sawtooth</b>	Framework	Modular blockchain suite
<b>Iroha</b>	Framework	Simple blockchain for mobile/web
<b>Burrow</b>	Framework	Ethereum Virtual Machine
<b>Caliper</b>	Tool	Blockchain performance benchmark
<b>Composer</b>	Tool	Business network development

**Categories:**

- **Frameworks:** Core blockchain platforms
- **Tools:** Development and testing utilities

**Mnemonic**

“F-S-I-B-C-C” (Fabric, Sawtooth, Iroha, Burrow, Caliper, Composer)

**Question 4(b) OR [4 marks]**

Explain Practical Byzantine Fault Tolerance algorithm in detail.

**Solution**

**Definition:** Consensus algorithm that works correctly even when up to  $1/3$  of nodes are faulty or malicious.

Table 20: PBFT Phases

Phase	Description	Purpose
<b>Pre-prepare</b>	Primary broadcasts request	Initiate consensus
<b>Prepare</b>	Nodes validate and broadcast	Verify proposal
<b>Commit</b>	Nodes commit to decision	Finalize agreement

**Algorithm Steps:**

1. Client sends request to primary replica
2. Primary broadcasts pre-prepare message
3. Backups send prepare messages if valid
4. After receiving  $2f+1$  prepares, send commit
5. Execute after receiving  $2f+1$  commits

**Key Properties:**

- **Safety:** Never produces inconsistent results
- **Liveness:** Eventually produces results
- **Fault Tolerance:** Works with  $f < n/3$  faulty nodes

**Mnemonic**

“Pre-Prepare-Commit” (3 Phases of PBFT)

**Question 4(c) OR [7 marks]**

“Eventual consistency is evident in the context of bitcoin.” Justify the given statement.

**Solution**

**Definition:** Eventual consistency means the system will become consistent over time, even if it's temporarily inconsistent.

**Bitcoin Implementation:**

Table 21: Bitcoin Consistency Mechanisms

Mechanism	Description	Purpose
<b>Chain Reorganization</b>	Replace shorter chain with longer	Maintain consensus
<b>Confirmation Delays</b>	Wait for multiple blocks	Increase certainty
<b>Fork Resolution</b>	Longest chain wins	Resolve conflicts

### Scenarios Demonstrating Eventual Consistency:

1. **Temporary Forks:** When two miners find blocks simultaneously
2. **Network Partitions:** Isolated nodes may have different views
3. **Double Spending Attempts:** Conflicting transactions in different blocks

### Resolution Process:

- **Mining Continues:** Miners build on their preferred chain
- **Longest Chain Rule:** Network adopts chain with most work
- **Automatic Convergence:** All nodes eventually agree

### Diagram: Fork Resolution

#### Mermaid Diagram (Code)

```
{Shaded}
{Highlighting}[]
graph LR
    A[Block N] --{-}{-}{ B[Block N+1a]}
    A --{-}{-}{ C[Block N+1b]}
    B --{-}{-}{ D[Block N+2a]}
    C --{-}{-}{ E[Dies {-} Shorter Chain]}
    D --{-}{-}{ F[Becomes Main Chain]}
{Highlighting}
{Shaded}
```

### Justification Points:

- **Probabilistic Finality:** Longer confirmation time = higher certainty
- **No Immediate Consistency:** New transactions aren't instantly final
- **Convergence Guarantee:** Network will eventually agree on single chain
- **Time-based Resolution:** Consistency improves with time

### Practical Implications:

- **Merchant Waiting:** Wait for confirmations before accepting payment
- **Exchange Policies:** Different confirmation requirements for different amounts
- **Risk Management:** Balance speed vs security based on transaction value

### Mnemonic

“Time-Brings-Consistency” (Eventual Consistency = Time + Convergence)

## Question 5(a) [3 marks]

Explain advantages of ERC 20.

### Solution

Table 22: ERC-20 Token Advantages

Advantage	Description
<b>Standardization</b>	Common interface for all tokens
<b>Interoperability</b>	Works with all Ethereum wallets/exchanges
<b>Liquidity</b>	Easy trading and exchange

### Key Benefits:

- **Developer Friendly:** Simple implementation standard
- **Market Adoption:** Widely supported across platforms
- **Smart Contract Integration:** Easy DeFi integration

### Mnemonic

“Standard-Interoperable-Liquid” (SIL)

Question 5(b) [4 marks]

Describe working mechanism of a smart-contract in detail.

Solution

Table 23: Smart Contract Workflow

Step	Description
Code Deployment	Contract uploaded to blockchain
Trigger Conditions	Predefined conditions monitored
Automatic Execution	Contract executes when conditions met
State Update	Blockchain state modified

Working Process:

1. **Development:** Write contract in Solidity/Vyper
2. **Compilation:** Convert to bytecode
3. **Deployment:** Upload to blockchain network
4. **Execution:** Triggered by transactions or events

Mnemonic

“Deploy-Trigger-Execute-Update” (DTEU)

Question 5(c) [7 marks]

What is smart-contract? Explain features and applications of smart-contract in detail.

Solution

**Definition:** Self-executing contracts with terms directly written into code, running on blockchain.

Table 24: Smart Contract Features

Feature	Description	Benefit
Autonomous	Executes without intermediaries	Cost reduction
Transparent	Code visible on blockchain	Trust building
Immutable	Cannot be changed once deployed	Security
Deterministic	Same input produces same output	Predictability

## Diagram: Smart Contract Architecture

### Mermaid Diagram (Code)

```
{Shaded}
{Highlighting}[]
graph LR
    A[Smart Contract] --> B[Trigger Conditions]
    B --> C[Automatic Execution]
    C --> D[State Changes]
    D --> E[Event Emissions]
    E --> F[External Calls]
    F --> G[Other Contracts]
{Highlighting}
{Shaded}
```

### Applications:

Table 25: Smart Contract Applications

Domain	Use Case	Example
<b>Finance</b>	Automated lending	DeFi protocols
<b>Insurance</b>	Claim processing	Flight delay insurance
<b>Supply Chain</b>	Product tracking	Food provenance
<b>Real Estate</b>	Property transfers	Automated escrow
<b>Gaming</b>	Digital assets	NFT marketplaces

### Advantages:

- **Efficiency:** Reduced processing time and costs
- **Trust:** No need for trusted third parties
- **Accuracy:** Eliminates human errors
- **Global Access:** Available 24/7 worldwide

### Limitations:

- **Immutability:** Difficult to fix bugs after deployment
- **Oracle Problem:** Need external data sources
- **Gas Costs:** Execution costs can be high
- **Complexity:** Requires technical expertise

### Development Considerations:

- **Security Audits:** Essential before deployment
- **Testing:** Extensive testing on testnets
- **Upgradability:** Design patterns for updates
- **Gas Optimization:** Minimize execution costs

### Mnemonic

“Auto-Transparent-Immutable-Deterministic” (ATID) for features

## Question 5(a) OR [3 marks]

Explain disadvantages of ERC20.

### Solution

Table 26: ERC-20 Token Disadvantages

Disadvantage	Description
<b>Limited Functionality</b>	Only basic token operations
<b>No Built-in Security</b>	Vulnerable to common attacks
<b>Gas Dependency</b>	Requires ETH for transactions

**Key Issues:**

- **Transfer Limitations:** Cannot handle complex transfers
- **Approval Risks:** Double spending vulnerabilities
- **Network Congestion:** High fees during peak times

**Mnemonic**

“Limited-Vulnerable-Dependent” (LVD)

**Question 5(b) OR [4 marks]**

Describe steps for Launching of a Decentralized Autonomous Organization (DAO)?

**Solution**

Table 27: DAO Launch Steps

Step	Description
<b>Concept Design</b>	Define purpose and governance rules
<b>Smart Contract Development</b>	Code governance mechanisms
<b>Token Distribution</b>	Allocate voting rights
<b>Community Building</b>	Attract members and contributors

**Detailed Process:**

1. **Whitepaper Creation:** Document vision and tokenomics
2. **Technical Implementation:** Deploy governance contracts
3. **Initial Funding:** Raise capital through token sales
4. **Operations Launch:** Begin decentralized operations

**Mnemonic**

“Design-Develop-Distribute-Deploy” (4D Launch)

**Question 5(c) OR [7 marks]**

What is Decentralized Autonomous Organization (DAO)? Explain its advantages and disadvantages in detail.

**Solution**

**Definition:** A blockchain-based organization governed by smart contracts and token holders rather than traditional management.

Table 28: DAO Structure

Component	Description	Function
<b>Smart Contracts</b>	Governance rules in code	Automated decision execution
<b>Tokens</b>	Voting rights and ownership	Democratic participation
<b>Proposals</b>	Suggested changes or actions	Community-driven initiatives
<b>Treasury</b>	Shared funds	Resource allocation

## Diagram: DAO Governance Flow

### Mermaid Diagram (Code)

```
{Shaded}
{Highlighting}[]
graph LR
    A[Token Holders] --> B[Submit Proposals]
    B --> C[Community Discussion]
    C --> D[Voting Period]
    D --> E[Execution if Passed]
    E --> F[Smart Contract Updates]
    F --> G[Treasury Actions]
{Highlighting}
{Shaded}
```

### Advantages:

Table 29: DAO Benefits

Advantage	Description	Impact
<b>Decentralization</b>	No single point of control	Reduced corruption risk
<b>Transparency</b>	All decisions on blockchain	Enhanced accountability
<b>Global Participation</b>	Anyone can join	Diverse perspectives
<b>Efficiency</b>	Automated execution	Faster decision implementation
<b>Democratic Governance</b>	Token-based voting	Fair representation

### Disadvantages:

Table 30: DAO Challenges

Disadvantage	Description	Risk
<b>Technical Complexity</b>	Smart contract bugs	System failures
<b>Legal Uncertainty</b>	Unclear regulatory status	Compliance issues
<b>Coordination Problems</b>	Difficult decision making	Slow progress
<b>Token Concentration</b>	Wealthy holders control votes	Centralization risk
<b>Security Vulnerabilities</b>	Code exploits possible	Financial losses

### Types of DAOs:

- **Investment DAOs:** Collective investment decisions
- **Protocol DAOs:** Blockchain protocol governance
- **Social DAOs:** Community-driven organizations
- **Collector DAOs:** NFT and art collecting

### Success Factors:

- **Clear Purpose:** Well-defined mission and goals
- **Robust Governance:** Effective voting mechanisms
- **Community Engagement:** Active member participation
- **Technical Security:** Audited smart contracts
- **Legal Compliance:** Regulatory compliance where applicable

### Notable Examples:

- **MakerDAO:** Decentralized finance protocol
- **Uniswap:** Decentralized exchange governance
- **Compound:** Money market protocol

### Future Outlook:

- **Regulatory Clarity:** Evolving legal frameworks
- **Technical Improvements:** Better governance tools
- **Mainstream Adoption:** Growing corporate interest
- **Integration:** Hybrid traditional-DAO models

### Mnemonic

“Decentralized-Autonomous-Organization” (DAO = Democratic Automated Ownership)