Module 3

Q. 1 Explain the types of smart contracts.

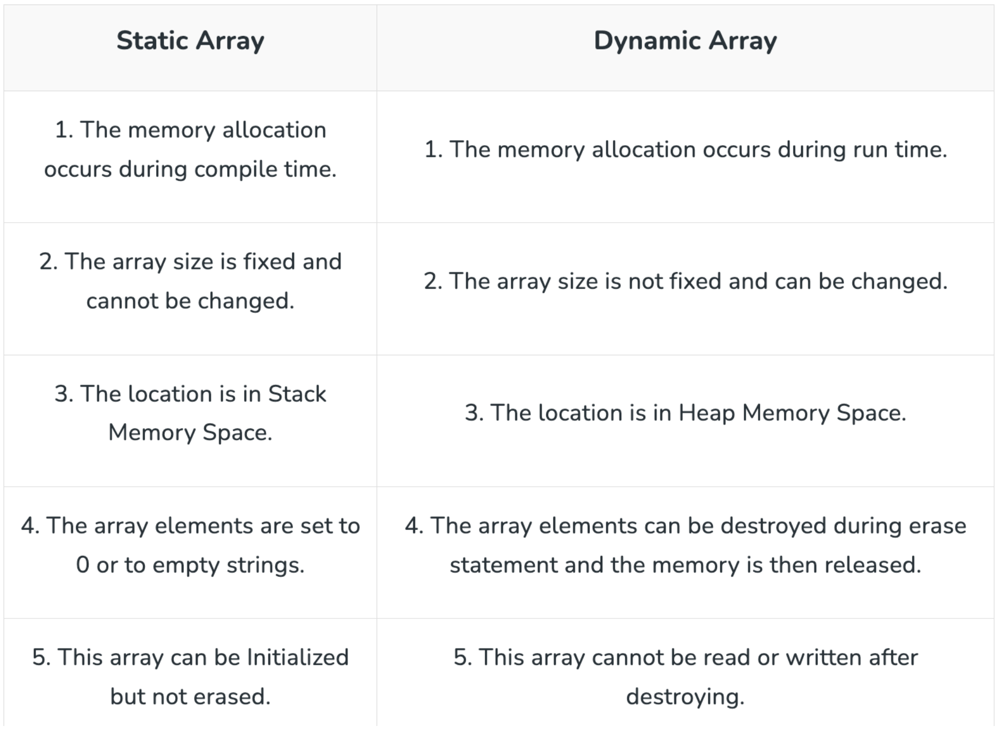
Smart contracts are simply programs stored on a blockchain that run when predetermined conditions are met. They typically are used to automate the execution of an agreement. There are three categories of such contracts – Smart Legal Contracts, Decentralized Autonomous Organizations, and Application Logic Contracts.

1. Smart Legal Contracts:
   These contracts are legally enforceable and require the parties to satisfy their contractual obligations. Parties may face strict legal actions if they fail to comply.

2. Decentralized Autonomous Organizations (DAO):
   a. For a DAO, the backbone is its smart contract. The contract is bound to specific rules that are coded into blockchain contracts blended with governance mechanisms.
   b. It has diverse use cases that range from simple to complex, which depends on the number of stakeholders.
   c. DAOs are open-source and also feature transparency, and, in theory, are incorruptible. Plus, any action taken by the community members gets replaced by a self-enforcing code.

3. Application Logic Contracts (ALC):
   a. Yet another type of smart contract in Blockchain is Application Logic Contracts (ALCs), which allow devices to function securely and autonomously. Plus, ALCs ensure greater automation, cheaper transactions, and scalability.
   b. These contracts contain an application-based code, which typically remains in sync with other blockchain contracts. It enables communication across different devices, such as the Internet of Things (IoT) merger with blockchain technology.

Q. 2 Differentiate between dynamic arrays and fixed arrays.



Static Array	Dynamic Array
1. The memory allocation occurs during compile time.	1. The memory allocation occurs during run time.
2. The array size is fixed and cannot be changed.	2. The array size is not fixed and can be changed.
3. The location is in Stack Memory Space.	3. The location is in Heap Memory Space.
4. The array elements are set to 0 or to empty strings.	4. The array elements can be destroyed during erase statement and the memory is then released.
5. This array can be Initialized but not erased.	5. This array cannot be read or written after destroying.

Q. 3 Write a solidity code on the given problem statement.

Q. 4 Explain error handling in smart contracts.

1. Earlier versions of Solidity utilized a single throw statement for error handling, but this method proved suboptimal for gas efficiency and required additional test functions to check values and throw errors.

2. To address this issue, Solidity version 4.10 introduced three special functions – assert, require, and revert – as new error handling constructs.

3. The Require Function in Solidity:
   The require function plays a pivotal role in developing secure and reliable Solidity contracts. By employing the require function effectively, you can validate inputs, enforce conditions, and enhance the overall integrity of your contract's logic.

4. The Assert Function in Solidity:
   The assert function is used to check for internal errors in the contract code. It takes a boolean expression as its argument and throws an exception and reverts the transaction if the expression evaluates to false. However, unlike the require function, the assert function should only be used to check for internal errors in the contract code that should never occur.

5. The Revert Function in Solidity:
   The revert function is similar to the require function in that it's used to revert a transaction if a condition is not met. However, the revert function provides more flexibility in error handling and allows you to provide a reason string to explain why the transaction was reverted.

Module 4

Q. 1 Explain Ethereum and components of Ethereum.

1. Mining Node: These nodes are responsible for writing all the transactions that have occurred in the Ethereum network in the block.
2. Ethereum Virtual Machine Node: These are the nodes in the Ethereum network in which Smart Contracts are implemented. By default, this node utilizes a 30303 port number for the purpose of communication among themselves.

3. Ether: Ether is a type of cryptocurrency used in the Ethereum network just like bitcoin is used in a blockchain network. It is a peer-to-peer currency, similar to Bitcoin. It tracks and promotes each transaction in the network.

4. Gas: Gas is an internal currency of the Ethereum network. We need gas to run applications on the Ethereum network, much as we need gas to run a vehicle.

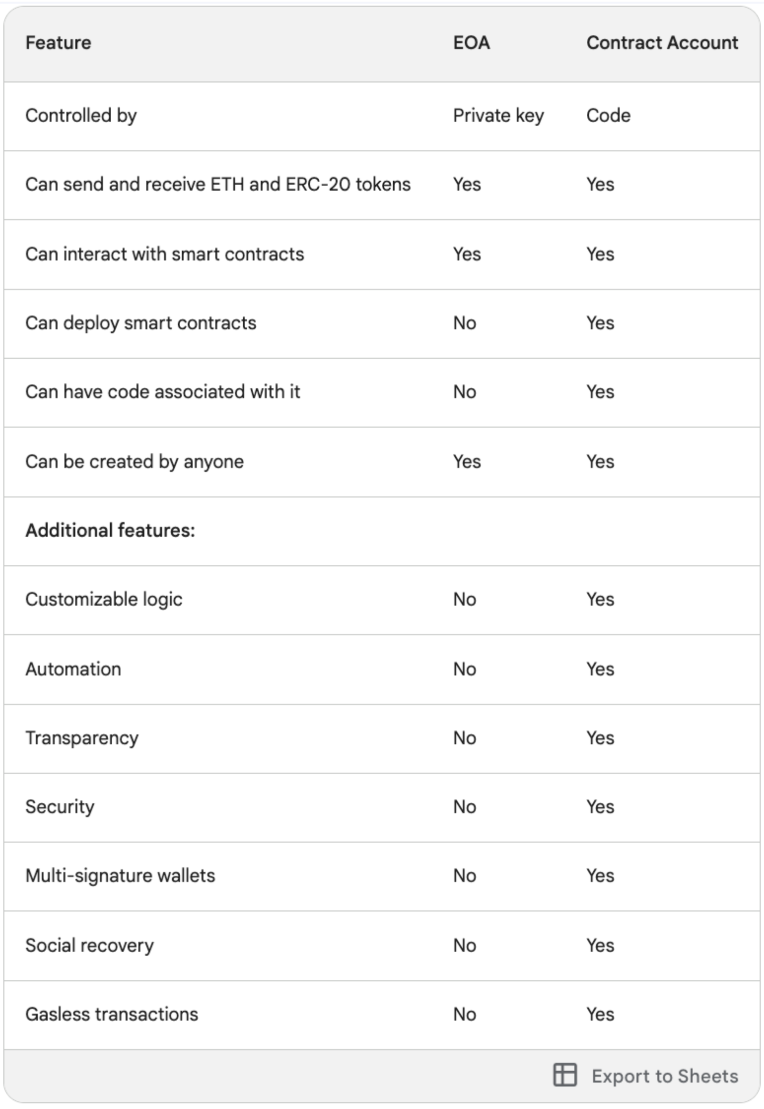
5. Ethereum Accounts: There are two types of Ethereum accounts. They are as follows.
   1. Externally owned account: These accounts are used to store transactions.
   2. Contract account: As the name itself suggests, these accounts store the details of Smart Contracts.

6. Ethash
   The intended PoW algorithm for Ethereum 1.0 is Ethash.

7. Transaction
   An Ethereum transaction refers to an action initiated by an externally-owned account, in other words an account managed by a human, not a contract. For example, if Bob sends Alice 1 ETH, Bob's account must be debited and Alice's must be credited. This state-changing action takes place within a transaction. Transactions, which change the state of the EVM, need to be broadcast to the whole network

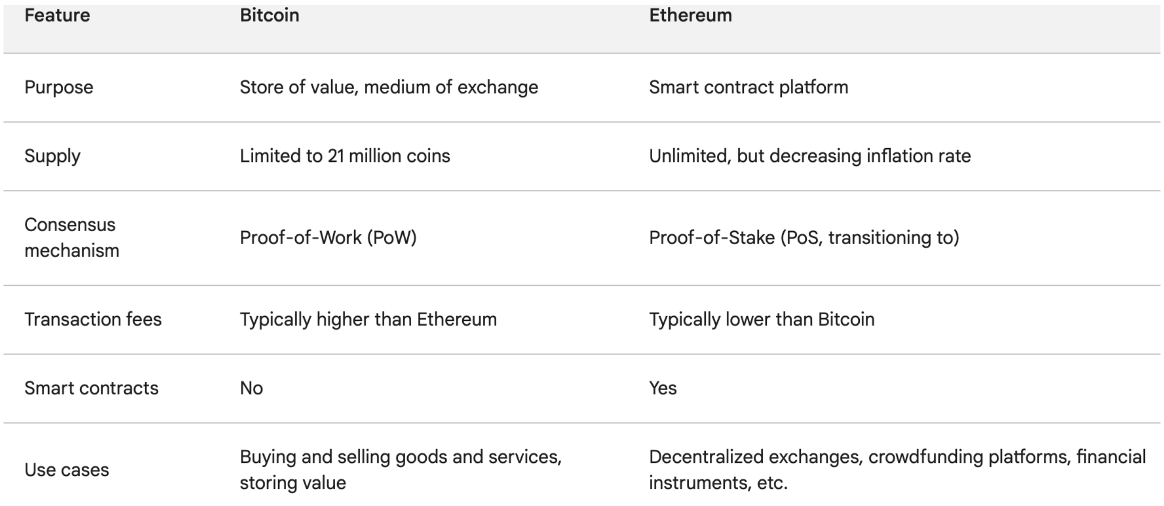
8. Swarm and Whisper
   Both Swarm and Whisper are complementary technologies contributing to the vision of Ethereum as a "world computer". When imagining Ethereum as a metaphor for a shared computer, it should be noted that computation alone is not enough. For a computer to be fully useful, it also needs storage to "remember" things and bandwidth to "communicate" them. This could be summarised as such:
   1. Contracts: decentralized logic
   2. Swarm: decentralized storage
   3. Whisper: decentralized messaging

Q. 2 Differentiate between EOA and Contract accounts.



Feature	EOA	Contract Account
Controlled by	Private key	Code
Can send and receive ETH and ERC-20 tokens	Yes	Yes
Can interact with smart contracts	Yes	Yes
Can deploy smart contracts	No	Yes
Can have code associated with it	No	Yes
Can be created by anyone	Yes	Yes
Additional features:		
Customizable logic	No	Yes
Automation	No	Yes
Transparency	No	Yes
Security	No	Yes
Multi-signature wallets	No	Yes
Social recovery	No	Yes
Gasless transactions	No	Yes

Q. 3 Compare Bitcoin and Ethereum.



Feature	Bitcoin	Ethereum
Purpose	Store of value, medium of exchange	Smart contract platform
Supply	Limited to 21 million coins	Unlimited, but decreasing inflation rate
Consensus mechanism	Proof-of-Work (PoW)	Proof-of-Stake (PoS, transitioning to)
Transaction fees	Typically higher than Ethereum	Typically lower than Bitcoin
Smart contracts	No	Yes
Use cases	Buying and selling goods and services, storing value	Decentralized exchanges, crowdfunding platforms, financial instruments, etc.

Q. 4 Short note on test networks in Ethereum.

Core advantage of testnet is ability to deploy your code and check for vulnerabilities, errors, and complete testing phase without any financial burden on developer and extra transactions on the mainnet. Below is a comparison of some of most popular testnets –

1. Ropsten :

    1. Proof of Work
    2. Not immune to spam attacks
    3. Supported by geth and parity
    4. Chain Id: 3
    5. Network Id: 3
    6. Block time: sub-30 seconds
    7. Faucet: Ropsten testnet is deprecated so, Faucet link is not working.

2. Kovan :

    1. Proof of Authority
    2. Immune to spam attacks
    3. Supported by geth and parity
    4. Chain Id: 42
    5. Network Id: 42
    6. Block time: 4 seconds
    7. Faucet: Kovan testnet is deprecated so, Faucet link is not working.

3. Rinkeby :

    1. Proof of Authority
    2. Immune to spam attacks
    3. Supported by geth and parity
    4. Chain Id: 4
    5. Network Id: 4
    6. Block time: 15 seconds
    7. Faucet: Rinkeby testnet is deprecated so, Faucet link is not working.

4. Sokol :

1. Proof of Authority
2. Immune to spam attacks
3. Supported by geth and parity
4. Chain Id: 77
5. Network Id: 77
6. Block time: 5 seconds
7. Faucet: The Sokol testnet is currently lacking validators so, Faucet link is not working.

5. Goerli :

1. Proof of Authority
2. Immune to spam attacks
3. Supported by multiple clients like Geth, Pantheon, Nethermind, and Parity
4. Chain Id: 5
5. Network Id: 5
6. Block time: 15 seconds on average
7. Faucet: https://goerlifaucet.com/

Module 5

Q. 1 Explain state machine replication?

1. State machine replication (SMR) is a technique for achieving fault tolerance in distributed systems by replicating the state of a system across multiple servers. SMR works by ensuring that all replicas of the system state are kept in sync, so that even if some servers fail, the system can continue to operate correctly.

2. SMR uses a consensus algorithm to ensure that all replicas of the system state agree on the same state.

3. Once the replicas of the system state are in sync, they can process client requests independently. Each replica will process the requests in the same order and produce the same outputs. This ensures that the system remains consistent, even if some replicas fail.

4. SMR is a powerful technique for achieving fault tolerance in distributed systems. It is used in a wide variety of applications, including databases, file systems, and messaging systems. Here is an example of how SMR can be used to implement a fault-tolerant database:

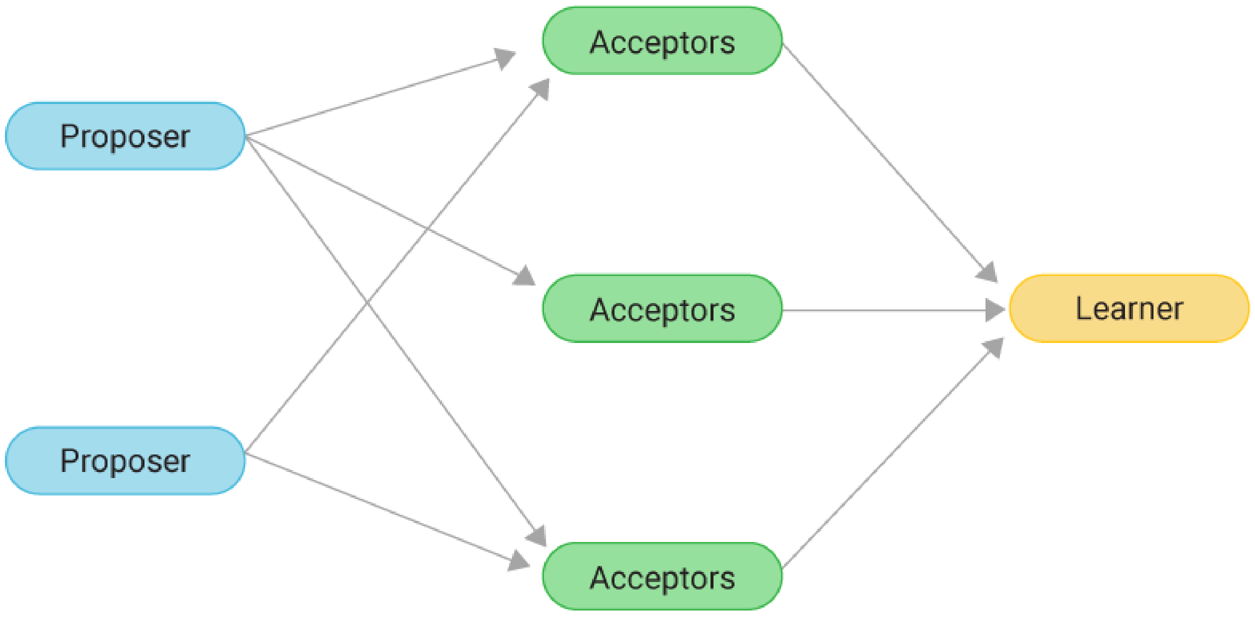
   a. The database is replicated on multiple servers.
   b. When a client makes a request to the database, the request is sent to all of the replicas.
   c. Each replica processes the request and produces a response.
   d. The replicas use a consensus algorithm to agree on a single response.
   e. The agreed-upon response is sent back to the client.

5. If a replica fails, the remaining replicas can continue to operate and process client requests. The system will remain consistent, because all of the replicas will agree on the same state.

Q. 2 Explain PAXOS consensus algorithm.
Paxos is an algorithm that enables a distributed set of computers to achieve consensus over an asynchronous network. To achieve agreement, one or more of the computers proposes a value to Paxos. Consensus is achieved when a majority of the computers running Paxos agrees on one of the proposed values.

In general terms, Paxos selects a single value from one or more of the values that are proposed, and then broadcasts that value to all of the cooperating computers. Once the Paxos algorithm has run, all of the computers (or database nodes)) agree upon the proposed value, and the cluster clocks forward.

Paxos defines several different roles which must cooperate to achieve consensus. They are:



1. Proposers, who receive requests (values) from clients and try to convince acceptors to accept the value they propose.
2. Acceptors, who accept certain proposed values from proposers and let proposers know whether a different value was accepted. A response from an acceptor represents a vote for a particular proposal.
3. Learners, who announce the outcome to all participating nodes.

It is important to note that this algorithm requires a majority of nodes to agree on a value for consensus to be achieved. This is to prevent a situation where two different values are agreed upon by different subsets of nodes

Q. 3 Explain Byzantine Fault Tolerant Consensus Protocol used in private blockchain.

1. Byzantine Fault Tolerant (BFT) consensus protocols are used in private blockchains to achieve consensus among nodes, even if some nodes are faulty or malicious.

2. This is important for private blockchains, which are often used in high-stakes environments where it is critical to prevent fraud and other malicious activity.

3. There are a number of different BFT consensus protocols that can be used in private blockchains. Some of the most popular protocols include:

   a. Practical Byzantine Fault Tolerance (PBFT): PBFT is a classic BFT consensus protocol that has been used in a variety of systems, including private blockchains. PBFT is a synchronous protocol, which means that it assumes that all nodes communicate with each other in a timely manner.

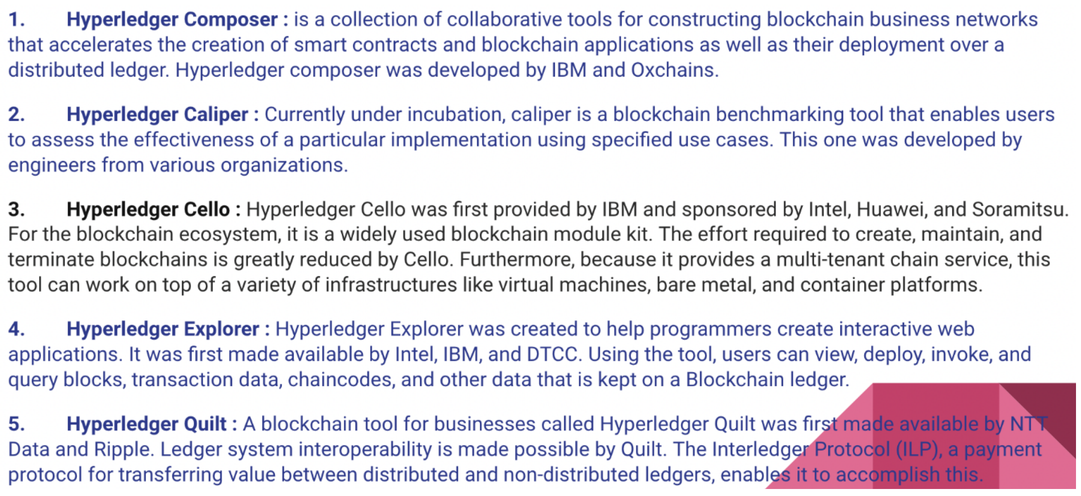
   b. Tendermint Consensus: Tendermint is a newer BFT consensus protocol that is specifically designed for blockchains. Tendermint is an asynchronous protocol, which means that it does not assume that all nodes communicate with each other in a timely manner. This makes Tendermint more suitable for use in large-scale distributed networks.

   c. Hyperledger Fabric: Hyperledger Fabric is a popular private blockchain platform that uses a unique BFT consensus protocol called Kafka BFT. Kafka BFT is a permissioned protocol that is based on the Kafka distributed streaming platform.

4. The specific BFT consensus protocol that is used in a private blockchain will depend on the specific needs of the network.

5. Here is an example of how a BFT consensus protocol might be used in a private blockchain:
   1. A node proposes a new block to the network.
   2. All other nodes verify the block and send their votes to the proposer.
   3. If a majority of nodes vote in favor of the block, it is added to the blockchain.
   4. If a majority of nodes vote against the block, it is rejected.
   5. This process is repeated for each new block that is proposed.

Q. 4 Write a short note on Hyperledger tools.



1. **Hyperledger Composer :** is a collection of collaborative tools for constructing blockchain business networks that accelerates the creation of smart contracts and blockchain applications as well as their deployment over a distributed ledger. Hyperledger composer was developed by IBM and Oxchains.

2. **Hyperledger Caliper :** Currently under incubation, caliper is a blockchain benchmarking tool that enables users to assess the effectiveness of a particular implementation using specified use cases. This one was developed by engineers from various organizations.

3. **Hyperledger Cello :** Hyperledger Cello was first provided by IBM and sponsored by Intel, Huawei, and Soramitsu. For the blockchain ecosystem, it is a widely used blockchain module kit. The effort required to create, maintain, and terminate blockchains is greatly reduced by Cello. Furthermore, because it provides a multi-tenant chain service, this tool can work on top of a variety of infrastructures like virtual machines, bare metal, and container platforms.

4. **Hyperledger Explorer :** Hyperledger Explorer was created to help programmers create interactive web applications. It was first made available by Intel, IBM, and DTCC. Using the tool, users can view, deploy, invoke, and query blocks, transaction data, chaincodes, and other data that is kept on a Blockchain ledger.

5. **Hyperledger Quilt :** A blockchain tool for businesses called Hyperledger Quilt was first made available by NTT Data and Ripple. Ledger system interoperability is made possible by Quilt. The Interledger Protocol (ILP), a payment protocol for transferring value between distributed and non-distributed ledgers, enables it to accomplish this.