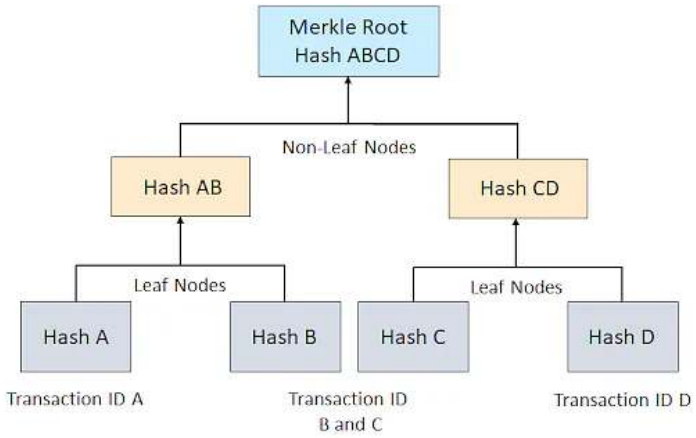
Chapter 1

Define Blockchain

- Blockchain is a list of records called blocks that store data publicly and in chronological order.
- This information is encrypted using cryptography to ensure that the privacy of the user is not compromised and data cannot be altered.
- Blockchain is a peer-to-peer (P2P) distributed network that connects nodes.
- Blockchain is decentralized. It means there's no central point of authority in the network.
- Each node carries the same copy of network records.
- Every network update (such as adding a new block) requires consensus or global network agreement.

Explain concept of Merkle tree and its use in Blockchain.

- Merkle tree is a mathematical data structure composed of hashes of different blocks of data, and which serves as a summary of all the transactions in a block.
- Merkle tree is a fundamental part of blockchain technology.
- It also allows for efficient and secure verification of content in a large body of data.
- Both Bitcoin and Ethereum use Merkle Trees structure.
- Merkle Tree is also known as Hash Tree.
- A Merkle tree stores all the transactions in a block by producing a digital fingerprint of the entire set of transactions.
- It allows the user to verify whether a transaction can be included in a block or not.
- Merkle Root is stored in the block header.



- Use in blockchain:
  - It is used to validate the data's integrity effectively.
  - Compared to other data structures, the Merkle tree takes up very little disk space, hence the size of records on each node is minimized.
  - Merkle trees can be broken down into small pieces of data for verification, hence making verification less computationally intensive.As the data format is efficient, verifying the data's integrity takes only a few moments.

Explain in detail the components of Blockchain.

Node

- Full Node: It maintains a full copy of all the transactions. It has the capacity to validate, accept and reject the transactions.
- Partial Node: It is also called a Lightweight Node because it doesn't maintain the whole copy of the blockchain ledger. It maintains only the hash value of the transaction. These nodes have low storage and low computational power.

Ledger

It is a digital database of information.

There are three types of ledger. They are:

- Public Ledger – It is open and transparent to all. Anyone in the blockchain network can read or write something.
- Distributed Ledger – In this ledger, all nodes have a local copy of the database. Here, a group of nodes collectively execute the job i.e verify transactions, add blocks in the blockchain.
- Decentralized Ledger – In this ledger, no one node or group of nodes has a central control. Every node participates in the execution of the job.

Wallet

It is a digital wallet that allows users to store their cryptocurrency.

Every node in the blockchain network has a Wallet.

Privacy of a wallet in a blockchain network is maintained using public and private key pairs.

Cryptocurrency wallets are mainly of two types –

- Hot Wallet – These wallets are used for online day-to-day transactions connected to the internet. Hackers can attack this wallet as it is connected to the internet. Hot wallets are further classified into two types –
    - Online/ Web wallets – These wallets run on the cloud platform. Examples – MyEtherWallet, MetaMask Wallet.
    - Software wallets – It consists of desktop wallets and mobile wallets. Desktop wallets can be downloaded on a desktop and the user has full control of the wallet. An example of a desktop wallet is Electrum.
    - Mobile wallets: designed to run on mobile devices: eg. mycelium.
- Cold Wallet – These wallets are not connected to the internet. It is very safe and hackers cannot attack it. These wallets are purchased by the user. Example – Paper wallet, hardware wallet.
    - Paper wallet – They are offline wallets in which a piece of paper is used that contains the crypto address. The private key is printed in QR code format. QR code is scanned for cryptocurrency transactions.
    - Hardware wallet – It is a physical electronic device that uses a random number generator that is associated with the wallet.

The focus of wallets is on these three things –

- Privacy
- Transactions should be secure

- Easy to use

Nonce

- A nonce is an abbreviation for “number only used once,” which is a number added to a hashed or encrypted block in a blockchain.
- It is the 32-bit number generated randomly only one time that assists to create a new block or validate a transaction.
- It is used to make the transaction more secure.
- It is hard to select the number which can be used as the nonce.
- It requires a vital amount of trial-and-error.
- First, a miner guesses a nonce. Then, it appends the guessed nonce to the hash of the current header.
- After that, it rehashes the value and compares this to the target hash.
- Now it checks whether the resulting hash value meets the requirements or not. If all the conditions are met, it means that the miner has created an answer and is granted the block.

Hash

The data is mapped to a fixed size using hashing. It plays a very important role in cryptography. In a blockchain network hash value of one transaction is the input of another transaction.
Properties of the hash function are as follows –

- Collision resistant
- Hiding
- Puzzle friendliness

Explain types of blockchain.

Public Blockchain

- These blockchains are completely open to following the idea of decentralization.
- They don’t have any restrictions, anyone having a computer and internet can participate in the network.
- As the name is public, this blockchain is open to the public - which means it is not owned by anyone.
- All the computer in the network hold the copy of other nodes or block present in the network

Advantages:

- Trustable: Public Blockchain nodes do not need to know or trust each other because the proof-of-work procedure ensures no fraudulent transactions.
- Secure: A public network can have as many participants or nodes as it wants, making it a secure network. The higher the network's size, the more records are distributed, and the more difficult it is for hackers to hack the entire network.
- Open and Transparent: The data on a public blockchain is transparent to all member nodes. Every authorized node has a copy of the blockchain records or digital ledger.

Disadvantages:

- Lower TPS: The number of transactions per second in a public blockchain is extremely low. This is because it is a large network with many nodes which take time to verify a transaction and do proof-of-work.
- Scalability Issues: Its transactions are processed and completed slowly. This harms scalability. Because the more we try to expand the network's size, the slower it will become.
- High Energy Consumption: The proof-of-work device is expensive and requires lots of energy.

Use Cases:

- Voting: Governments can use a public blockchain to vote, ensuring openness and trust.
- Public Blockchain is secured with proof of work or proof of stake; they can be used to displace traditional financial systems.

Examples of public blockchain are Bitcoin, Ethereum.

Private Blockchain

- This blockchain operates in a private context, such as a restricted network, or is controlled by a single identity.
- While it has a similar peer-to-peer connection and decentralization to a public blockchain network, this Blockchain is far smaller.
- Only selected nodes can participate in the process, making it more secure than the others.
- These blockchains are operated in a closed network.

Advantages:

- Speed: Private Blockchain transactions are faster. This is because a private network has a smaller number of nodes, which shortens the time it takes to verify a transaction.
- Scalability: You can tailor the size of your private Blockchain to meet your specific requirements. This makes private blockchains particularly scalable since they allow companies to easily raise or decrease their network size.
- Privacy: It has increased the level of privacy for confidentiality reasons as the businesses required.

Disadvantages:

- Trust Building: In a private network, there are fewer participants than in a public network.
- Lower Security: A private blockchain network has fewer nodes or members, so it is more vulnerable to a security compromise.
- Centralization: Private blockchains are limited in that they require a central Identity and Access Management (IAM) system to function.

Use Cases:

- Supply Chain Management: A private blockchain can be used to manage a company's supply chain.
- Asset Ownership: A private blockchain can be used to track and verify assets.
- Internal Voting: Internal voting is also possible with a private blockchain.

Examples of Private Blockchain: R3 (Banks), EWF (Energy)

Hybrid Blockchain

- Organizations who expect the best of both worlds use a hybrid blockchain, which combines the features of both private and public blockchains.
- It enables enterprises to construct a private, permission-based system alongside a public, permissionless system, allowing them to choose who has access to certain Blockchain data and what data is made public.
- In a hybrid blockchain, transactions and records are typically not made public, but they can be validated if necessary by granting access via a smart contract.

Advantages of Hybrid Blockchain -

- Secure: Hybrid Blockchain operates within a closed environment, preventing outside hackers from launching a 51 percent attack on the network.
- Cost-Effective: It also safeguards privacy while allowing third-party contact. Transactions are inexpensive and quick and scale better than a public blockchain network.

Disadvantages of Hybrid Blockchain -

- Lack of Transparency: Because information can be hidden, this type of blockchain isn't completely transparent.
- Less Incentive: Upgrading can be difficult, and users have no incentive to participate in or contribute to the network.

Uses of Hybrid Blockchain -

- Real Estate: Real-estate companies can use hybrid networks to run their systems and offer information to the public.
- Retail: The hybrid network can also help retailers streamline their processes.
- Highly Regulated Markets: Hybrid blockchains are also well-suited to highly regulated areas like the banking sector.

Examples of Hybrid Blockchain: IBM Food Trust

Consortium Blockchain

- In the same way that a hybrid blockchain has both private and public blockchain features, a Consortium blockchain, also known as a federated blockchain, does.
- However, it differs because it involves various organizational members working together on a decentralized network.
- Predetermined nodes control the consensus methods in a consortium blockchain.

- It has a validator node responsible for initiating, receiving, and validating transactions. Transactions can be initiated or received by member nodes.

Advantages of Consortium Blockchain -

- Secure: A consortium blockchain is more secure, scalable, and efficient than a public blockchain network. It, like private and mixed blockchains, has access controls.

Disadvantages of Consortium Blockchain -

- Lack of Transparency: The consortium blockchain has a lower degree of transparency. If a member node is infiltrated, it can still be hacked, and the Blockchain's rules can render the network inoperable.

Uses of Consortium Blockchain -

- Banking and Payments: A consortium can be formed by a group of banks working together. They have control over which nodes will validate transactions.
- Research: A consortium blockchain can be employed to share research data and outcomes.
- Food Tracking: It is also apt for food tracking.

Examples of consortium Blockchain are Tendermint and Multichain.

Explain Pros and cons of blockchain

Pros:

- Immutability: Blockchain supports immutability, meaning it is impossible to erase or replace recorded data. Therefore, the blockchain prevents data tampering within the network.
- Transparency: Blockchain is decentralized, meaning any network member can verify data recorded into the blockchain. Therefore, the public can trust the network.
- Censorship: Blockchain technology is free from censorship since it does not have control of any single party. Therefore, no single authority (including governments) can interrupt the operation of the network.
- Traceability: Blockchain creates an irreversible audit trail, allowing easy tracing of changes on the network.

Cons:

- Speed and performance: Blockchain is considerably slower than the traditional database because blockchain technology carries out more operations.
- High implementation cost: Blockchain is costlier compared to a traditional database.
- Data modification: Blockchain technology does not allow easy modification of data once recorded, and it requires rewriting the codes in all of the blocks, which is time-consuming and expensive.

Explain life of miners in blockchain

- Anybody can apply for a miner, and you could run the client yourself.

- However, these miners use very powerful computers that are specifically designed to mine crypto transactions.
- They do this by actually solving math problems and resolving cryptographic issues because every transaction needs to be cryptographically encoded and secured.
- These mathematical problems ensure that nobody is tampering with that data.
- The cryptocurrency is created by rewarding these miners for their work in solving the mathematical and cryptographical problems.

Explain distributed ledger

- A distributed ledger is a database that is spread across various computers, nodes, institutions, or countries accessible by multiple people around the globe.
- Distributed ledgers use independent nodes to record, share, and synchronize transactions in their respective electronic ledgers instead of keeping them in one centralized server.
- A blockchain uses several technologies including distributed ledger technology to enable blockchain applications.
- Features:
    - Decentralized: It is a decentralized technology and every node will maintain the ledger, and if any data changes happen, the ledger will get updated.
    - Immutable: Distributed ledger uses cryptography to create a secure database in which data once stored cannot be altered or changed.
    - Append only: Distributed ledgers are append-only in comparison to the traditional database where data can be altered.
    - Distributed: In this technology, there is no central server or authority managing the database, which makes the technology transparent.
    - Shared: The distributed ledger is not associated with any single entity. It is shared among the nodes on the network where some nodes have a full copy of the ledger while some nodes have only the necessary information that is required to make them functional and efficient.

Explain Hashing function

A hash function is a mathematical function that takes an input string of any length and converts it to a fixed-length output string.
The fixed-length output is known as the hash value.
To be cryptographically secure and useful, a hash function should have the following properties:

- Collision resistant: Given two messages m1 and m2, it is difficult to find a hash value such that hash(k, m1) = hash(k, m2) where k is the key value.
- Preimage resistance: Given a hash value h, it is difficult to find a message m such that h = hash(k, m).
- Second preimage resistance: Given a message m1, it is difficult to find another message m2 such that hash(k, m1) = hash(k, m2).
- Large output space: The only way to find a hash collision is via a brute force search, which requires checking as many inputs as the hash function has possible outputs.

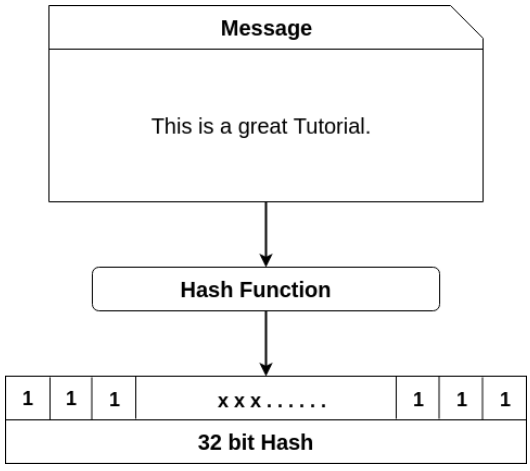
- Deterministic: A hash function must be deterministic, which means that for any given input a hash function must always give the same result.
- Avalanche Effect: This means for a small change in the input, the output will change significantly.
- Puzzle Friendliness: This means even if one gets to know the first 200 bytes, one cannot guess or determine the next 56 bytes.
- Fixed-length Mapping: For any input of fixed length, the hash function will always generate the output of the same length.

What is the difficulty? Explain the requirement and use.

- Cryptocurrency difficulty is a measure of how difficult it is to mine a block in a blockchain for a particular cryptocurrency.
- A high cryptocurrency difficulty means it takes additional computing power for mining.
- Cryptocurrency difficulty is a parameter that bitcoin and other cryptocurrencies use to keep the average time between blocks steady as the network's hash power changes.
- Cryptocurrency difficulty is important since a high difficulty can help secure the blockchain network against malicious attacks.
- Without such a system in place, blocks would likely be discovered faster and faster as more miners joined the network with increasingly sophisticated equipment.
- This would result in new bitcoin entering circulation at an unpredictable rate and would likely have the knock-on effect of inhibiting its rise in value.

How hash is calculated in blockchain

- A cryptographic hash is a digest or digital fingerprint of a certain amount of data.
- The hash function takes the input of variable lengths and returns outputs of fixed lengths.
- In cryptographic hash functions, the transactions are taken as inputs and the hash algorithm gives an output of a fixed size.
- SHA-256 is the most famous of all cryptographic hash functions because it’s used extensively in blockchain technology.



- The hash algorithm has certain unique properties:
    - It produces a unique output (or hash).

- It is a one-way function.

Differentiate between blockchain and database

Database	Blockchain
Database uses centralized storage of data.	Blockchain uses decentralized storage of data.
Database needs a Database admin or Database administrator to manage the stored data.	There is no administrator in Blockchain.
Modifying data requires permission from database admin.	Modifying data does not require permission. Users have a copy of data and by modifying the copies does not affect the master copy of the data as Blockchain is irresistible to modification of data.
Centralized databases keep information that is up-to-date at a particular moment	Blockchain keeps the present information as well as the past information that has been stored before.
Centralized databases are used as databases for a really long time and have a good performance record, but are slow for ertain functionalities.	Blockchain is ideal for transaction platform but it slows down when used as databases, specially with large collection of data.

Differentiate between open and close blockchain.

Open blockchain	**Closed blockchain**
It is a public network that maintains an immutable record of transactions.	It is a private network that maintains a shared record of transactions.
Anyone can publish a transaction and participate in the network by adhering to a set of published rules.	The network is accessible only to those who have permission and transactions can be edited by administrators.
Eg. bitcoin, litecoin	Eg. Hyperledger, Ripple
Public open: Public and open blockchains are available for everybody and written data is accessible and readable by everybody as well.	Public closed: Public and closed blockchains are open to everyone to write, but only restricted people can read the data.
Public and open blockchains support use	A use case for this kind of blockchain for

cases such as public/transparent ledgers where everybody can read and verify data.	example is voting or polling. Everybody can write his/her vote or opinion to the blockchain, but only the creators of the ballot box are allowed to read the voting results.
Public open blockchain example is bitcoin	Public and closed blockchains are often used for medical, legal or financial use cases where customers or prospects can store confidential and / or personal information for restricted access by the corresponding entities.
Private and Open: This type of private blockchain is commonly used in supply chains, where only suppliers are able to write the supply status to the chain, but every private blockchain's participant can track the status and see the information.	Private and Closed: Private and closed blockchains enable use cases where only trusted and known members are able to write and read the data in the blockchain (e.g. an interbank blockchain where banks exchange assets).

Table 1: Main types of blockchains segmented by permission model

Blockchain types			Read	Write	Commit	Example
	Open	*Public permissionless*	Open to anyone	Anyone	Anyone*	Bitcoin, Ethereum
		Public permissioned	Open to anyone	Authorised participants	All or subset of authorised participants	Sovrin
	Closed	*Consortium*	Restricted to an authorised set of participants	Authorised participants	All or subset of authorised participants	Multiple banks operating a shared ledger
		Private permissioned ('enterprise')	Fully private or restricted to a limited set of authorised nodes	Network operator only	Network operator only	Internal bank ledger shared between parent company and subsidiaries

** Requires significant investment either in mining hardware (proof-of-work model) or cryptocurrency itself (proof-of-stake model).*

Differentiate between public and private blockchain.

Basis of comparison	Public blockchain	Private blockchain
Access	In this type of blockchain anyone can read, write and participate in a blockchain.	In this type of blockchain read and write is done upon invitation.
Permission	It is a permissionless blockchain. It is public to everyone.	It is a permissioned blockchain.
Network actors	Don’t know each other	Know each other
Decentralized vs centralized	Decentralized	Centralized
Native token	Yes	Not necessary
Speed	Slow	Fast
Transactions	TPS are lesser	TPS is more as compared
Security	More secure due to decentralization and active participation	More prone to hacks, risks and data breaches / manipulation.
Energy consumption	Consumes more energy	Consumes a lot less energy
Consensus algorithm	Proof of Work, Proof of Stake, Proof of Burn, Proof of Space	Proof of Elapsed Time, Raft and Istanbul BFT
Examples	Ethereum, Litecoin, Bitcoin	R3 (banks), EWF (Energy)

Differentiate between permissionless and permissioned blockchain

	Permissionless	Permissioned
OVERVIEW	Open network available for anyone to interact and participate in consensus validation. Fully decentralized across unknown parties.	Closed network. Designated parties interact and participate in consensus validation. Partially decentralized (i.e., distributed across known parties).
ALSO KNOWN AS	Public, trustless.	Private, permissioned sandbox.
KEY ATTRIBUTES	■ Full transparency of transactions, based on open source protocols ■ Development via open source ■ Mostly anonymous, with some exceptions ■ Privacy depends on technological limitations or innovations ■ No central authority ■ Often involves digital asset or token for incentives	■ Controlled transparency, based on organizations' goals ■ Development via private entities ■ Not anonymous ■ Privacy depends on governance decisions ■ No single authority, but a private group authorizes decisions ■ May or may not involve digital assets or tokens
BENEFITS	■ **Broader decentralization**, extending access across more network participants ■ **Highly transparent**, which is beneficial for speed and reconciliation across unknown parties ■ **Censorship resistant**, due to accessibility and participation across locations and nationalities ■ **Security resilience**, since attackers cannot target a single repository, and it is costly and difficult to corrupt 51% of the network	■ **Incremental decentralization**, but participation from multiple businesses helps mitigate risks of highly centralized models ■ **Stronger information privacy** because transaction information is only available based on permissions ■ **Highly customizable** to specific use cases through diverse configurations, modular components and hybrid integrations ■ **Faster and more scalable**, since fewer nodes manage transaction verification and consensus
PITFALLS	■ **Less energy efficient** because network-wide transaction verification is resource-intensive ■ **Slower and difficult to scale**, as high volume can strain network-wide transaction verifications ■ **Less user privacy** and information control	■ **Limited decentralization** because a network with fewer participants increases risk of corruption or collusion ■ **Risk of override**, since owners and operators can control or change the rules of consensus, immutability, or mining ■ **Less transparent to outside oversight**, since participants are limited and operators determine privacy requirements
MARKET TRACTION	■ Peer-to-peer ■ Business-to-consumer ■ Government-to-citizens	■ Business-to-business ■ Business-to-consumer ■ Governments-to-organizations

Image link

Define following

- **Miners:** Miners validate new blockchain transactions and record them on the blockchain. Miners compete to solve a difficult mathematical problem based on a cryptographic hash algorithm.
- **Mining:** Mining is the process that Bitcoin and several other cryptocurrencies use to generate new coins and verify new transactions. It involves vast, decentralized networks of computers around the world that verify and secure blockchains.
- **Nonce:** It is a four-bit (32 byte) number added to a hashed—or encrypted—block in a blockchain that, when rehashed, meets the difficulty level restrictions. The nonce is the number that blockchain miners are solving for.
- **Hash tree:** A hash tree encodes the blockchain data in an efficient and secure manner. It enables the quick verification of blockchain data, as well as quick movement of large amounts of data from one computer node to the other on the peer-to-peer blockchain network.

Chapter 2

Differentiate between bitcoin and ethereum

BITCOIN VERSUS ETHEREUM

Basis of comparison	Bitcoin	Ethereum
Creator	Satoshi Nakamoto	Vitalik Buterin
Currency Issuance	12.5 bitcoins every 10 minutes	3 ethers every 15 seconds
Currency Cap	21 million coins	No cap
Block Creation	1 block every 10 minutes	1 block every 15 seconds
Scripting Language	Limited inbuilt language	Multi-purpose turing complete integrated language
Costing	Based on size	Based on operations and storage
Block Limits	1 MB & 8MB bitcoin cash	Gas limits
Accounts	One account	Two accounts
Algorithm Performance	Uses special Hardware	Can't use special hardware
Future Plans	Not clearly defined	-Change algorithm to POS -Use of sharding process

DB Difference Between.net

What is consensus? Why is it required in blockchain?

- Consensus for blockchain is a procedure in which the peers of a Blockchain network reach agreement about the present state of the data in the network. Through this, consensus algorithms establish reliability and trust in the Blockchain network.
- Consensus mechanisms (also known as consensus protocols or consensus algorithms) are used to verify transactions and maintain the security of the underlying blockchain
- There are many different types of consensus mechanisms, each with various benefits and drawbacks
- Proof of work (PoW) and proof of stake (PoS) are two of the most widely used consensus mechanisms
- Why Blockchains Need Consensus Mechanisms:

- Consensus mechanisms form the backbone of all cryptocurrency blockchains, and are what make them secure.
- In order to guarantee that all participants (‘nodes’) in a blockchain network agree on a single version of history, blockchain networks like Bitcoin and Ethereum implement what’s known as consensus mechanisms (also known as consensus protocols or consensus algorithms). These mechanisms aim to make the system fault-tolerant.

Explain different consensus algorithm

- The proof of work (PoW) is a common consensus algorithm used by the most popular cryptocurrency networks like bitcoin and litecoin. It requires a participant node to prove that the work done and submitted by them qualifies them to receive the right to add new transactions to the blockchain. However, this whole mining mechanism of bitcoin needs high energy consumption and a longer processing time.
- The proof of stake (PoS) is another common consensus algorithm that evolved as a low-cost, low-energy consuming alternative to the PoW algorithm. It involves the allocation of responsibility in maintaining the public ledger to a participant node in proportion to the number of virtual currency tokens held by it. However, this comes with the drawback that it incentivizes cryptocoin hoarding instead of spending.
- While PoW and PoS are by far the most prevalent in the blockchain space, there are other consensus algorithms like Proof of Capacity (PoC) which allow sharing of memory space of the contributing nodes on the blockchain network. The more memory or hard disk space a node has, the more rights it is granted for maintaining the public ledger.
- Proof of Activity (PoA), used on the Decred blockchain, is a hybrid that makes use of aspects of both PoW and PoS.
- Proof of Burn (PoB) is another that requires transactors to send small amounts of cryptocurrency to inaccessible wallet addresses, in effect "burning" them out of existence.
- Another, called Proof of History (PoH), developed by the Solana Project and similar to Proof of Elapsed Time (PoET), encodes the passage of time itself cryptographically to achieve consensus without expending many resources.

Differentiate between hard and soft fork

Hard fork	**Soft fork**
Backwards incompatible	backwards compatible
Reversible in nature	Irreversible in nature
Created two blockchains at the end	Soft fork combines the split into one at the end
Hard fork can valid previously entered invalid transactions and vice-versa.	Soft fork can only invalidate previously entered valid transactions.

More secure and private	Less secure and private
Requires all miners to validate new rules	Soft fork can work well with the majority of miners
Changes the entire rules of a blockchain	Used to add new functions and properties to a blockchain
Alters the characteristics of a blockchain	Modifies the characteristics of a blockchain
Does not require a soft fork for reversible process	Requires hard fork for reversible process

What is a Wallet? Explain types of Wallet. Differentiate between hot and cold wallets.

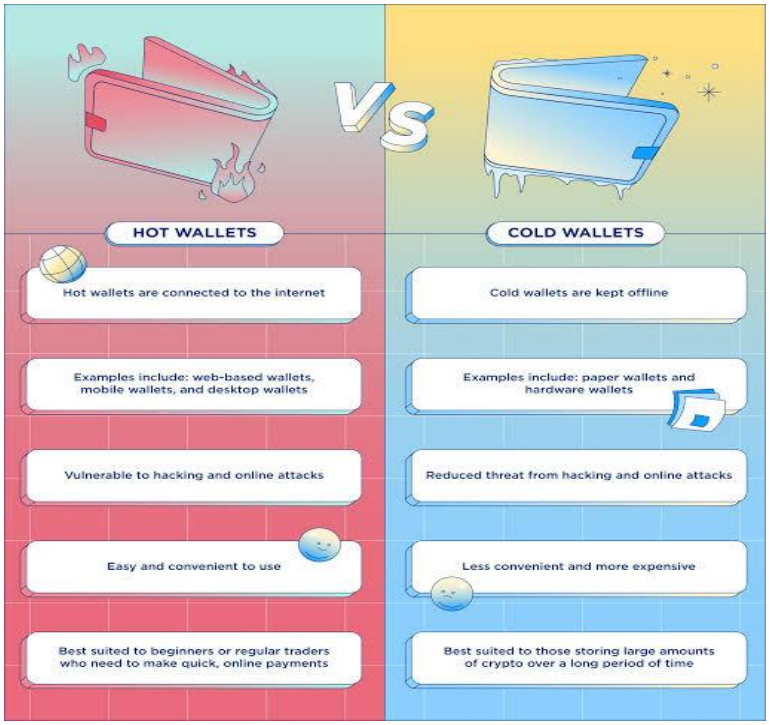
The letter 'E' in e-wallet stands for electronic; thus, e-wallet refers to an electronic wallet. An e-wallet, also known as a digital wallet, enables users to conduct transactions with ease. These wallets can be used for simple transactions both online and in stores.

Types of wallet are as follows:

- **Closed wallet**
  - A closed wallet enables users to make payments via an app or website.
  - They are typically created by businesses that sell products or services to their customers.
  - Users of a closed wallet can only use the stored funds to complete a transaction with the wallet's issuer.
  - If a transaction is canceled or a refund is issued, the entire amount is stored in the wallet.
- **Semi-closed wallet**
  - A semi-closed wallet allows users to easily make transactions at merchants and locations that are listed.
  - Semi-closed wallets have a limited coverage area.
  - To accept payment from the wallet, merchants must accept the contract or agreement with the issuer.
- **Open wallet**
  - Banks provide open wallets.
  - Users with open wallets can use them for any type of transaction.
  - Open wallets allow you to easily transfer funds.
  - Payments can be made both online and in-store at any time.
  - The open e-wallets service provider allows users to conduct transactions from anywhere in the world; however, both the sender and the receiver must have accounts on the same application.
- **Crypto wallet**
  - Users' public and private keys are stored in cryptocurrency wallets.
  - The keys could be cryptocurrency ownership certificates.

    - Hardware wallets, also known as cold wallets, add an extra layer of security and safety.
    - A USB stick can be used to operate wallets offline.
    - These wallets can be used to make cryptocurrency payments.
- **IoT wallet**
- IoT is an abbreviation for the Internet of Things.
- These are installed in watches, jackets, wristbands or other wallet enabled devices such as smart car computers, smart refrigerators, and others.
- IoT wallets work with e-money and virtual currencies.

Difference between a hot wallet and cold wallet.



Explain different types of cryptocurrencies.

A cryptocurrency is a form of digital currency that can be used to verify the transfer of assets, control the addition of new units, and secure financial transactions using cryptography.

Types of Cryptocurrency :

- Bitcoin
- Altcoin
    - Derived from bitcoin
    - Derived from original blockchain
- Token

Bitcoin

- Bitcoin is the original blockchain cryptocurrency.

- Most popular
- Widely used
- Derived from bitcoin[fork]: Namecoin, Peercoin, litecoin

Altcoin/Coins [Alternative Cryptocurrency Coins]

- Refers to a group of cryptocurrencies, ultimately all the cryptocurrencies other than Bitcoin.
- Most of them are forks of bitcoin and Ethereum.
- They are created to improve or add additional features or security like improved block time, transaction management, scripting language,consensus mechanism.
- Eg. Ripple, Tether, Bitcoin Cash, Bitcoin SV

Tokens

- Crypto tokens are a type of cryptocurrency that represents an asset or specific use and reside on their own blockchain.
- Tokens can be used for investment purposes, to store value, or to make purchases.
- Cryptocurrencies are digital currencies used to facilitate transactions (making and receiving payments) along the blockchain.

Explain UTXO

In bitcoin, the transaction lives until it has been executed till the time another transaction is done out of that UTXO. UTXO stands for Unspent Transaction Output.

- It is the amount of digital currency someone has left remaining after executing a transaction.
- When a transaction is completed, the unspent output is deposited back into the database as input which can be used later for another transaction.
- UTXOs are created through the consumption of existing UTXOs. Every Bitcoin transaction is composed of inputs and outputs. Inputs consume an existing UTXO, while outputs create a new UTXO.
- The UTXO model does not incorporate wallets at the protocol level. It is based on individual transactions that are grouped in blocks. The UTXO model is a design common to many cryptocurrencies, most notably Bitcoin.
- Cryptocurrencies that use the UTXO model do not use accounts or balances. Instead, UTXOs are transferred between users much like physical cash.
- Each transaction in the UTXO model can transition the system to a new state, but transitioning to a new state with each transaction is infeasible.
- The network participants must stay in sync with the current state.

Importance of UTXO Model :

- Language Agnostic Smart Contracts: The UTXO based smart contracts are independent of language and allow UTXOs to develop unique consensus mechanisms.
- Support for Decentralized Exchanges and Atomic Swap: The UTXO model could support atomic swaps, hence enabling peer-to-peer crypto trades without the involvement of a third party. The atomic swap feature of UTXOs offers a better facility for direct cryptocurrency trades between user wallets.
- Scalability Benefits: Facility or parallel transaction processing reduces computation load on the blockchain networks.
- Privacy and Security: With new addresses used for every UTXO transaction, it is impossible to track the transactions.
- Prevents Double Spending: A UTXO can only be used once, this is fundamental within the operation of the blockchain technology that guarantees that the currencies are not used more than once.
- More Flexibility: It provides more flexibility than fiat currency.
- Simple parallelization: It allows for the simpler parallelization of transactions in the smart contracts.

Refer for example

Chapter 3

Solidity programming

Solidity in 2 mins (for beginners) or **Solidity in 100 Seconds**

Solidity is an object-oriented programming language for implementing smart contracts on various blockchain platforms, most notably, Ethereum.
File Extension is .sol

Example:

```
pragma solidity >=0.4.0 <0.6.0;
contract SimpleStorage {
   uint storedData;
   function set(uint x) public {
      storedData = x;
   }
   function get() public view returns (uint) {
      return storedData;
   }
}
```

Pragma:
The first line of any solidity program is a pragma directive which tells the source code's intended Solidity version.

Solidity Contract:
A Solidity contract is a collection of functions and states that resides at a specific address on the Ethereum blockchain.

Function:
Function in Solidity is defined by using the 'function' keyword, followed by a unique function name, a list of parameters, scope, data type of returned value and a statement block surrounded by curly braces.

```
// function definition
function function-name(parameter-list) scope returns() {
   //statements
}
// function call
function-name()
```

Solidity supports three types of variables.

- State Variables − Variables whose values are permanently stored in a contract storage.
- Local Variables − Variables whose values are present till function is executing.
- Global Variables − Special variables exist in the global namespace used to get information about the blockchain.

Value Types

- Boolean: This data type accepts only two values True or False.
- Integer: This data type is used to store integer values, int and uint are used to declare signed and unsigned integers respectively.
- Fixed Point Numbers: They can be declared as fixed and unfixed for signed and unsigned fixed-point numbers of varying sizes respectively.
- Address: Address holds a 20-byte value which represents the size of an Ethereum address. It can be used to get balance or to transfer a balance by balance and transfer method respectively.
- Bytes and Strings: Bytes are used to store a fixed-sized character set while the string is used to store the character set equal to or more than a byte. Byte has an advantage that it uses less gas, so better to use when we know the length of data.
- Enums: It is used to create user-defined data types, used to assign a name to an integral constant which makes the contract more readable, maintainable, and less prone to errors. Options of enums can be represented by unsigned integer values starting from 0.
- Arrays: An array is a collection of values of the same data type where each value has a particular location known as an index. By using the index location, the desired value can be accessed or modified. The array size can be fixed or dynamic.
- Struct: Solidity allows users to create and define their own type in the form of structures. The structure is a group of different types which can contain both value type and reference type.
- Mapping: Mapping is a reference type that stores the data in a key-value pair where a key can be any value type. It is like a hashtable or dictionary as in any other programming language, where values can be retrieved using corresponding key.

Inheritance:

Solidity supports inheritance between smart contracts, where multiple contracts can be inherited into a single contract. The contract from which other contracts inherit features is known as a base contract, while the contract which inherits the features is called a derived contract. Simply, they are referred to as parent-child contracts.

Types of inheritance supported:

- Single Inheritance
- Multi-level Inheritance
- Hierarchical Inheritance
- Multiple Inheritance

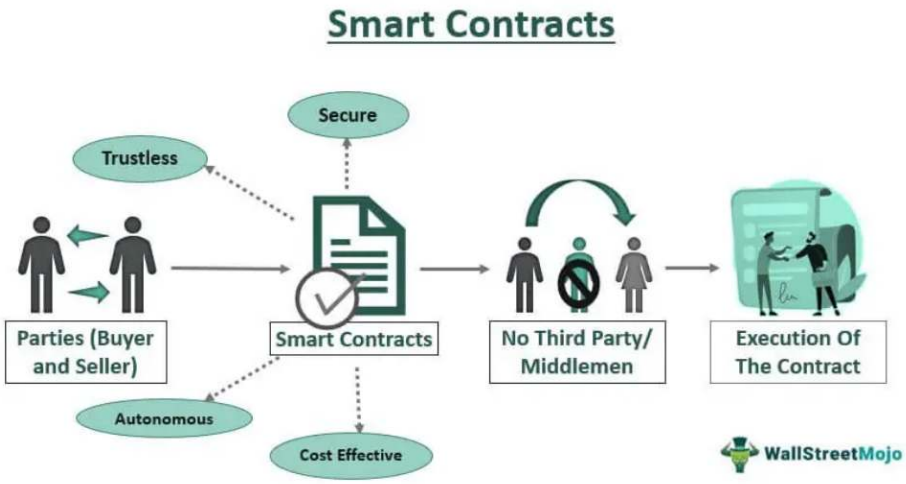
Error Handling

Errors can occur at compile time or runtime. Solidity is compiled to byte code and there a syntax error check happens at compile-time, while runtime errors are difficult to catch and occurs mainly while executing the contracts. Solidity has many functions for error handling. Some of the runtime errors are out-of-gas error, data type overflow error, divide by zero error, array-out-of-index error, etc.

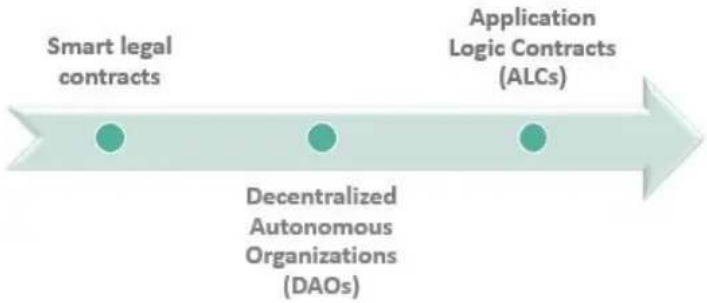
After version 4.10 new error handling construct assert, require, and revert statements were introduced.

Explain smart contract and its types

Smart contracts are digital transaction protocols that verify, control, and self-execute an agreement, embedded in computerized codes on a blockchain, if parties meet predefined rules. Unlike traditional (physical) ones, these contracts occur among anonymous parties and are enforced automatically without the involvement of any third party.



There are three types of self-executing contracts based on their applications:



- **Smart Legal Contracts:** These contracts are legally enforceable and require the parties to fulfill their contractual obligations. Failure to do so may result in strict legal actions against them.
- **Decentralized Autonomous Organizations:** These are blockchain communities that are bound to specific rules coded into blockchain contracts combined with governance mechanisms. Hence, any action taken by the community members gets replaced by a self-enforcing code.
- **Application Logic Contracts:** These contracts contain an application-based code that remains in sync with other blockchain contracts. It enables communication across different devices, such as the merger of the Internet of Things with blockchain technology.

Benefits of smart contract

- Autonomy and savings: Smart contracts do not need brokers or other intermediaries to confirm the agreement; thus, they eliminate the risk of manipulation by third parties. Moreover, the absence of intermediaries in smart contracts results in cost savings.
- Backup: All the documents stored on the blockchain are duplicated multiple times; thus, originals can be restored in the event of any data loss.
- Safety: Smart contracts are encrypted, and cryptography keeps all the documents safe from infiltration.
- Speed: Smart contracts automate tasks by using computer protocols, saving hours for various business processes.
- Accuracy: Using smart contracts eliminates errors that occur due to the manual filling of numerous forms.

How do smart contracts work??

- Smart contracts work by following simple “if/when…then…” statements that are written into code on a blockchain.
- A network of computers executes the actions when predetermined conditions have been met and verified.
- These actions could include releasing funds to the appropriate parties, registering a vehicle, sending notifications, or issuing a ticket.
- The blockchain is then updated when the transaction is completed.
- That means the transaction cannot be changed, and only parties who have been granted permission can see the results.
- Within a smart contract, there can be as many stipulations as needed to satisfy the participants that the task will be completed satisfactorily.
- To establish the terms, participants must determine how transactions and their data are represented on the blockchain, agree on the “if/when...then…” rules that govern those transactions, explore all possible exceptions, and define a framework for resolving disputes.
- Then the smart contract can be programmed by a developer – although increasingly, organizations that use blockchain for business provide templates, web interfaces, and other online tools to simplify structuring smart contracts.

What are blockchain oracles

- Blockchain oracles are entities that connect blockchains to external systems, thereby enabling smart contracts to execute based upon inputs and outputs from the real world
- Oracles provide a way for the decentralized Web3 ecosystem to access existing data sources, legacy systems, and advanced computations.
- Decentralized oracle networks (DONs) enable the creation of hybrid smart contracts, where on-chain code and off-chain infrastructure are combined to support advanced decentralized applications (dApps) that react to real-world events and interoperate with traditional systems.

- For example, let's assume Alice and Bob want to bet on the outcome of a sports match.
- Alice bets $20 on team A and Bob bets $20 on team B, with the $40 total held in escrow by a smart contract.
- When the game ends, how does the smart contract know whether to release the funds to Alice or Bob?
- The answer is it requires an oracle mechanism to fetch accurate match outcomes off-chain and deliver it to the blockchain in a secure and reliable manner.
- Solving the oracle problem is of the utmost importance because the vast majority of smart contract use cases like DeFi require knowledge of real-world data and events happening off-chain.

Advantages and disadvantages of smart contracts

Advantages:

- **Autonomy:** Smart contracts do not need brokers or other intermediaries to confirm the agreement; thus, they eliminate the risk of manipulation by third parties.
- **Savings**: The absence of intermediaries in smart contracts results in cost savings.
- **Backup**: All the documents stored on blockchain are duplicated multiple times; thus, originals can be restored in the event of any data loss.
- **Safety**: Smart contracts are encrypted, and cryptography keeps all the documents safe from infiltration.
- **Speed**: Smart contracts automate tasks by using computer protocols, saving hours of various business processes.
- **Accuracy**: Using smart contracts results in the elimination of errors that occur due to manual filling of numerous forms.

Disadvantages:

- **Difficult to change:** Changing smart contract processes is almost impossible, any error in the code can be time-consuming and expensive to correct.
- **Possibility of loopholes**: According to the concept of good faith, parties will deal fairly and not get benefits unethically from a contract. However, using smart contracts makes it difficult to ensure that the terms are met according to what was agreed upon.
- **Third party**: Although smart contracts seek to eliminate third-party involvement, it is not possible to eliminate them. Third parties assume different roles from the ones they take in traditional contracts. For example, lawyers will not be needed to prepare individual contracts; however, they will be needed by developers to understand the terms to create codes for smart contracts.
- **Vague terms:** Since contracts include terms that are not always understood, smart contracts are not always able to handle terms and conditions that are vague.

Explain features of solidity programming

- **Syntax**: Solidity syntax is similar to that of JavaScript.
- **File extension**: The file extension for Solidity is .sol
- **Applications**: Solidity is used to create contracts for various purposes such as voting, crowdfunding, blind auctions, multi-signature wallets, etc.

- **Contracts**: Solidity contracts are written in a format similar to class declarations in object-oriented languages. A contract can contain state variables, functions, and modifiers.
    - State variables can be of any type, including user-defined types.
    - Functions can be called externally or internally (within the contract).
    - Modifiers can be used to change the behavior of functions.
    - Contracts can also interact with each other, allowing for the creation of complex decentralized applications.
- **Data types**: Solidity provides several built-in types, including address, uint (unsigned integer), int (signed integer), bool (boolean), and byte (a fixed-size byte array). Solidity also allows for creating user-defined types, such as enums and structs.
- **Contract deployment**: Solidity contracts can be deployed on the Ethereum blockchain. Once deployed, a contract can be interacted with by any party that has the contract's address.
- **Solidity compiler**: The language is not yet fully standardized, and no official reference compiler exists. Solidity Compiler is maintained by the Ethereum Foundation.
- **Easy to learn:** Solidity is a high-level language, meaning it is easier to learn and use than a lower-level language like EVM bytecode. Several resources are available to help learn Solidity, including the official Solidity documentation.
- **Smart contracts**: Smart contracts are programs that run on the Ethereum blockchain and can be used to facilitate transactions and agreements between parties. Smart contracts can be used to create decentralized applications (d-apps).
- **DEXes**: Solidity is also used to create decentralized exchanges (DEXes). DEXes are exchanges that allow for the trading of cryptocurrency without the need for a central authority.
- **Programming component**s: The Solidity programming language has several high-level language components, which can be used to build contracts and programs. These include
    - Data types.
    - Functions.
    - Control structures.
    - Inheritance.
    - Packages.
    - Error handling.

Types of variables with example

Solidity supports three types of variables.

- State Variables − Variables whose values are permanently stored in a contract storage.
- Local Variables − Variables whose values are present till function is executing.
- Global Variables − Special variables exist in the global namespace used to get information about the blockchain.

State Variable

Variables whose values are permanently stored in a contract storage.

```
pragma solidity ^0.5.0;
contract SolidityTest {
   uint storedData;      // State variable
   constructor() public {
      storedData = 10;   // Using State variable
   }
}
```

Local Variable

Variables whose values are available only within a function where it is defined. Function parameters are always local to that function.

```
pragma solidity ^0.5.0;
contract SolidityTest {
   uint storedData; // State variable
   constructor() public {
      storedData = 10;
   }
   function getResult() public view
returns(uint){
      uint a = 1; // local variable
      uint b = 2;
      uint result = a + b;
      return result; //access the local
variable
   }
}
```

Global Variables

These are special variables which exist in global workspace and provide information about the blockchain and transaction properties.

Name	Returns
blockhash(uint blockNumber) returns (bytes32)	Hash of the given block - only works for 256 most recent, excluding current blocks
block.coinbase (address payable)	Current block miner's address
block.difficulty (uint)	Current block difficulty
block.gaslimit (uint)	Current block gaslimit
block.number (uint)	Current block number
block.timestamp (uint)	Current block timestamp as seconds since unix epoch

Explain the fallback function used in solidity

- The solidity fallback function is executed if none of the other functions match the function identifier or no data was provided with the function call.
- Only one unnamed function can be assigned to a contract and it is executed whenever the contract receives plain Ether without any data.
- To receive Ether and add it to the total balance of the contract, the fallback function must be marked payable.
- If no such function exists, the contract cannot receive Ether through regular transactions and will throw an exception.

Properties of a fallback function:

- Has no name or arguments.
- If it is not marked payable, the contract will throw an exception if it receives plain ether without data.
- Can not return anything.
- Can be defined once per contract.
- It is also executed if the caller meant to call a function that is not available
- It is mandatory to mark it external.
- It is limited to 2300 gas when called by another function. It is so as to make this function call as cheap as possible.

Explain address type in solidity

- An address value type is specifically designed to hold up to 20B, or 160 bits, which is the size of an Ethereum address.
- Solidity actually offers two address value types: address and address payable. The difference between the two is that addresses payable can send and transfer Ether.
- We can use an address to acquire a balance using the .balance method and to transfer a balance using the .transfer method.

Here's an example of an address data type in Solidity:

```
pragma solidity ^0.5.10

// example of an address value type in solidity
Contract  SampleAddress   }
  address public  myAddress =
0xb794f5ea0ba39494ce839613fffba74279579268;
}
```

Reference type in solidity

- Solidity reference types differ from value types in that they do not store values directly on their own.

- Instead, reference types store (or “reference”) the address of the data’s location and do not directly share the data.
- Reference data types must be handled cautiously since they deal with storage locations.
- Data location affects the amount of gas used in a transaction and therefore can negatively impact smart contract development performance.
- When using reference types, reference variables point a user to the location of value storage, and these can take up more than 32B of memory in size.
- Two separate variables can refer to the exact location, and any change in one variable can affect the other.
- Several variables that point to the same address can be used to effect a change in a reference data type.
- There are several reference data types in Solidity: fixed-sized arrays, dynamic-sized arrays, array members, byte arrays, string arrays, structs, and mapping.

Function declaration in solidity

- In Solidity a function is generally defined by using the function keyword, followed by the name of the function which is unique and does not match with any of the reserved keywords.
- A function can also have a list of parameters containing the name and data type of the parameter.
- The return value of a function is optional but in solidity, the return type of the function is defined at the time of declaration.

```
function function_name(parameter_list) scope returns(return_type) {
        // block of code
}
```

```
// Solidity program to demonstrate
// function declaration
pragma solidity ^0.5.0;

// Creating a contract
contract Test {

   // Defining function to calculate sum of 2 numbers
   function add() public view returns(uint){
      uint num1 = 10;
      uint num2 = 16;
      uint sum = num1 + num2;
      return sum;
   }
}
```

Error handling in solidity

Solidity provides various functions for error handling. Generally when an error occurs, the state is reverted back to its original state.

Other checks are to prevent unauthorized code access. Following are some of the important methods used in error handling −

- **assert(bool condition)** − In case condition is not met, this method call causes an invalid opcode, and any changes done to state got reverted. This method is to be used for internal errors.
- **require(bool condition)** − In case the condition is not met, this method call reverts to the original state. - This method is to be used for errors in inputs or external components.
- **require(bool condition, string memory message)** − In case condition is not met, this method call reverts to the original state. - This method is to be used for errors in inputs or external components. It provides an option to provide a custom message.
- **revert()** − This method aborts the execution and reverts any changes done to the state.
- **revert(string memory reason)** − This method aborts the execution and reverts any changes done to the state. It provides an option to provide a custom message.

Inheritance in Solidity

- It is a way of extending the functionality of a program, used to separate the code, reduces the dependency, and increases the re-usability of the existing code.
- Solidity supports inheritance between smart contracts, where multiple contracts can be inherited into a single contract.
- The contract from which other contracts inherit features is known as a base contract, while the contract which inherits the features is called a derived contract.

Types of inheritance in solidity

- Single Inheritance: In Single or single level inheritance the functions and variables of one base contract are inherited to only one derived contract.
- Multi-level Inheritance: It is very similar to single inheritance, but the difference is that it has levels of the relationship between the parent and the child. The child contract derived from a parent also acts as a parent for the contract which is derived from it.
- Hierarchical Inheritance: In Hierarchical inheritance, a parent contract has more than one child contract. It is mostly used when a common functionality is to be used in different places.
- Multiple Inheritance: In Multiple Inheritance, a single contract can be inherited from many contracts. A parent contract can have more than one child while a child contract can have more than one parent.
  examples here