Q.B. I

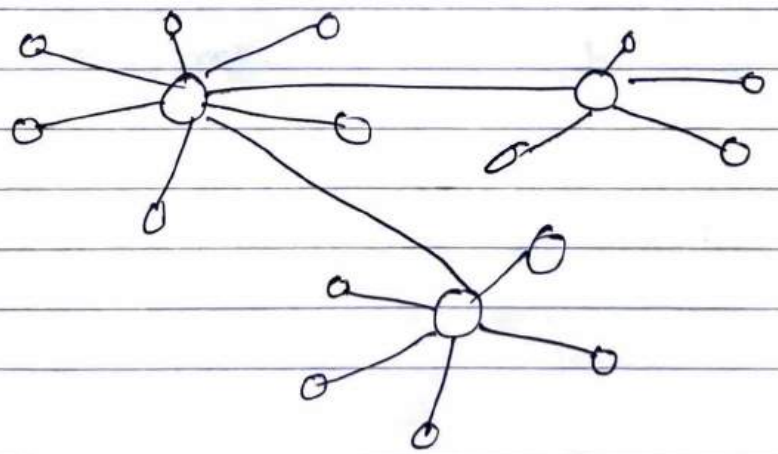
Q.1 Write short note on:

a) blockchain technology

⇒
- Blockchain technology is a decentralized digital ledger system used to record transactions across multiple computers in such a way that the registered transactions cannot be altered respectively.
- It ensures transparency, security and immutability of data through cryptographic technique.
- One of the famous blockchain is Bitcoin.
- Bitcoin is a cryptocurrency and is used to exchange digital asset online.
- Bitcoin uses cryptographic proof instead of third-party trust for two parties to execute transactions over the Internet.
- Each transaction protects through digital signature.

- Decentralization

⇒ There is no central server or system which keeps the data of blockchain.



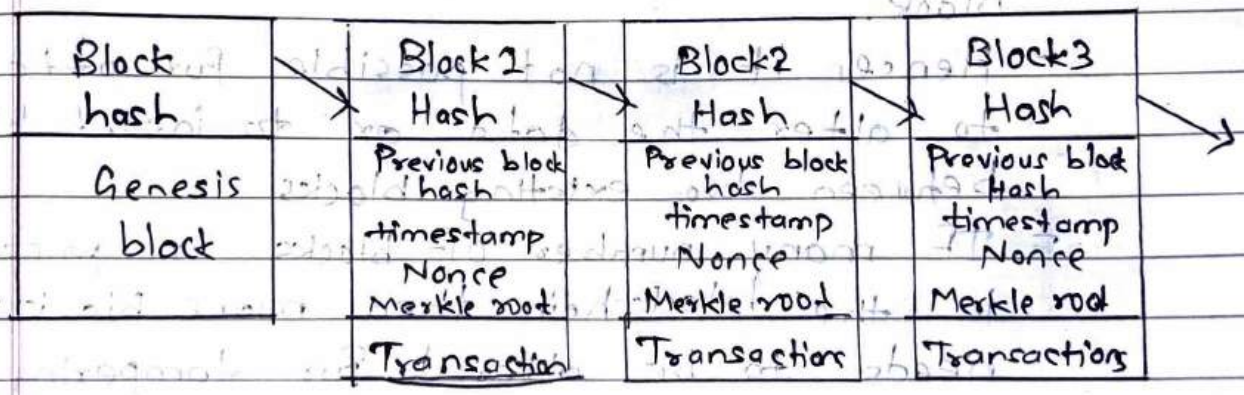
- The data is distributed over the Millions of Computers around the World which are connected to the blockchain.
- This system allows the Notarization of Data as it is present on every node and is publicly verifiable.

b) block in a blockchain

⇒

- Block in blockchain contains many transactions and following are the details that block comprises

1) Hash of block
2) Hash of previous block
3) Timestamp
4) Nonce
5) Merkle root.
6) Transaction data containing details of transactions in block.



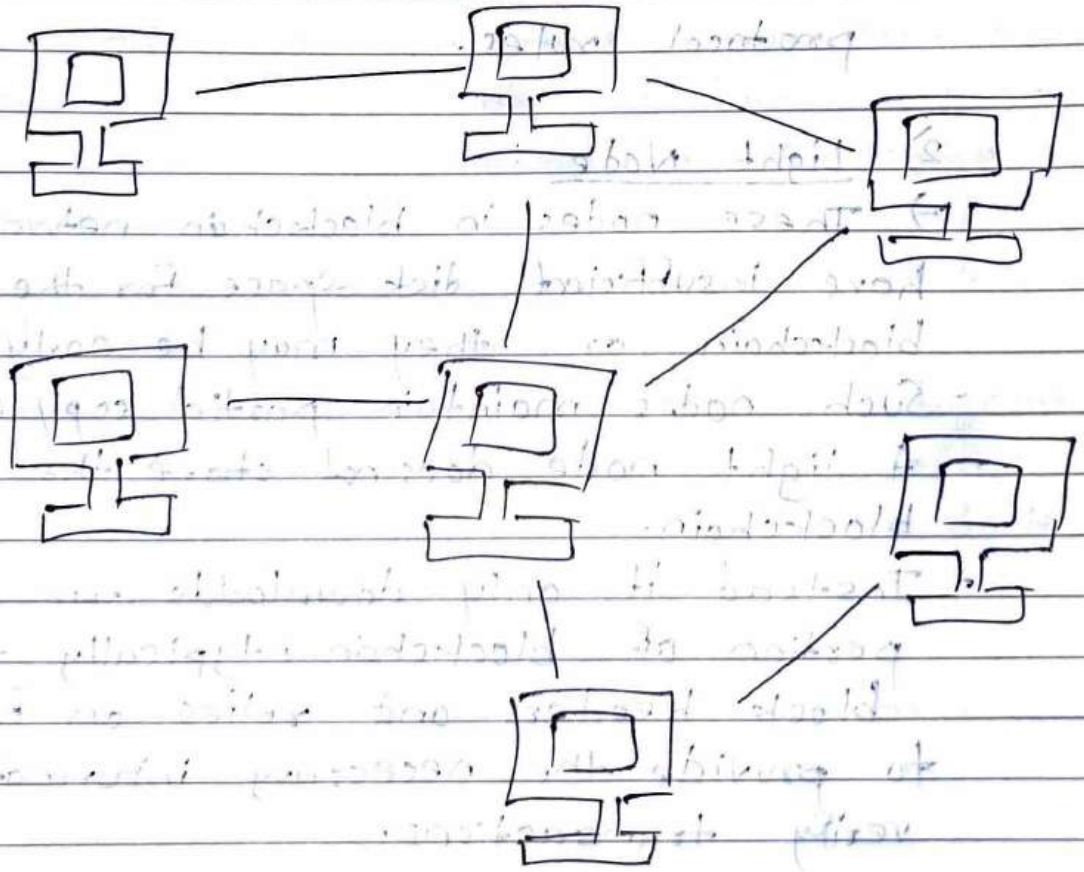
- Genesis block is very first block which does not contain data hash of previous block.
- Hence, previous block hash value in Genesis block is set to zero.
- This block includes transactions that are contained and validated to generate unique hash.

- The hash of genesis block is added to all new transactions in a new block.
- This combination is used to create its unique hash.
- This process is repeated until the new blocks are added to a blockchain.
- For example,
  In general block n hash is created with the hash of block n-1 and another set of new transactions.
- Hence, each block is linked with previous block via its hash.
- From any block, there is continuous linkage in reverse direction till genesis block.
- Hence, it is not possible for attacker to alter the data or to insert block between two existing blocks
- If many number of blocks are present in the blockchain then more blocks needs to be altered for tampering the data.

Q.2 What is node in blockchain? Define full node and light node.

⇒

- Any device such as computer, mobile device that is connected to the blockchain network is called as a node.
- These interconnected nodes have copy of blockchain ledger.
- In blockchain technology peer-to-peer architecture, every node maintains complete updated copy of blockchain ledger or database.
- Nodes maintains copy of blockchain and preserve Integrity of the Blockchain.



(Blockchain Nodes)

1) A Full node

⇒ This node keeps whole copy of transaction history of the blockchain.

- These nodes processes and accepts the blocks or transaction, validates them and then broadcast to Blockchain network.
- This is known as forging or mining.
- Nodes run the full node to help synch the blockchain.
- Full nodes :

i) Validates and relay transactions and blocks
ii) Provide increased security and decentralization to the network.
iii) Ensures transactions and blocks follow the protocol rules.

2) Light Node :

⇒ These nodes in blockchain network may have insufficient disk space for the full blockchain or they may be early users.

- Such nodes maintain partial copy of ledger.
- A light node does not store the entire blockchain.
- Instead it only downlodds some small portion of blockchain, typically the block headers and relies on full nodes to provide the necessary information to verify transactions.

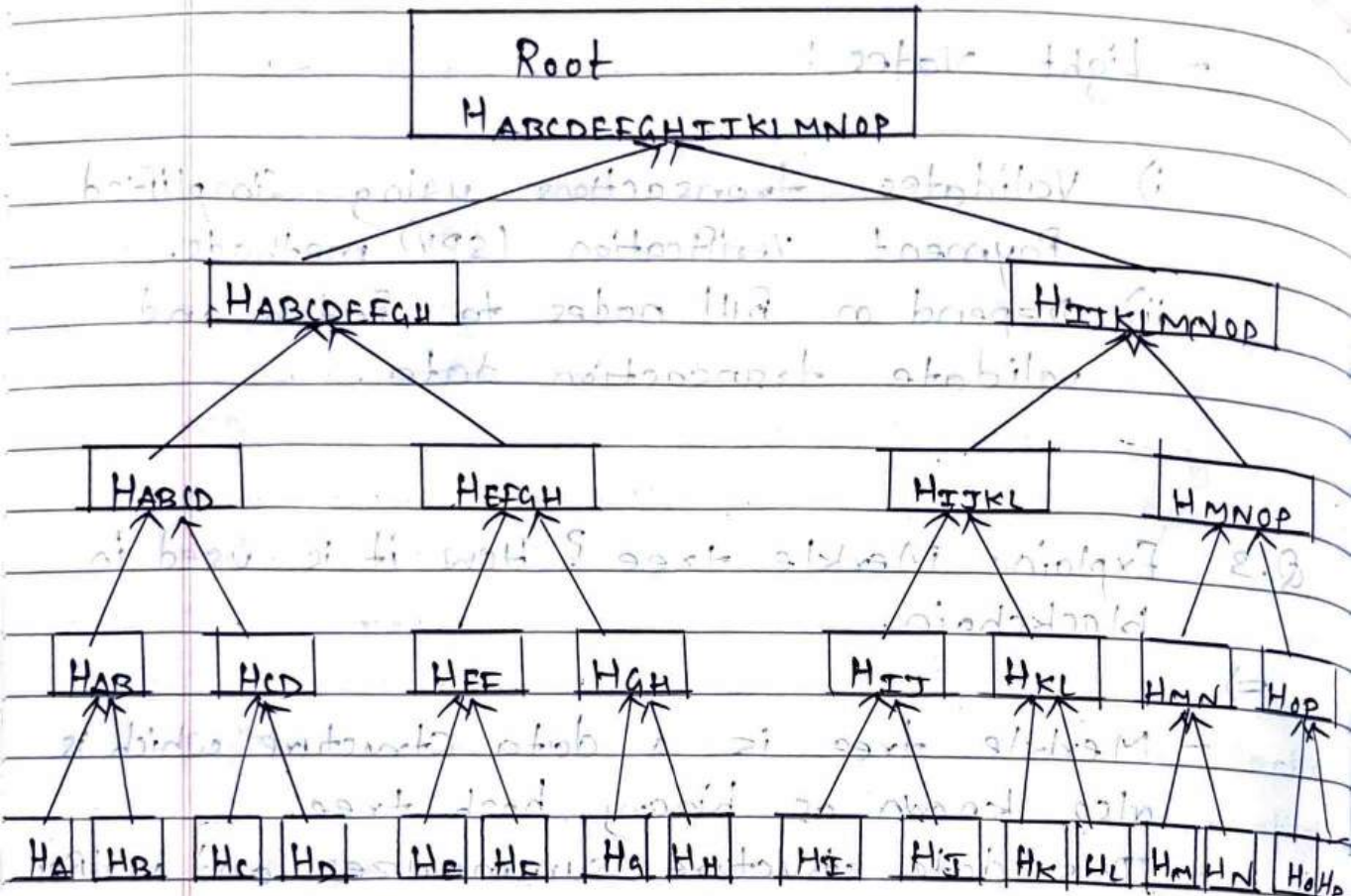
- Light Nodes :

i) Validates transactions using Simplified Payment Verification (SPV) methods.
ii) Depend on full nodes to fetch and validate transaction data.

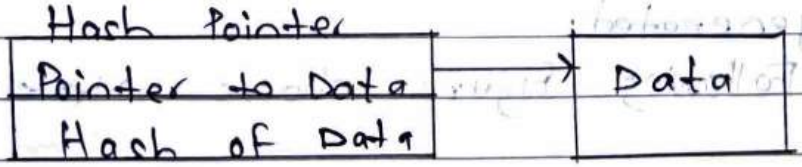
Q.3 Explain Merkle tree ? How it is used in blockchain.

⇒

- Merkle tree is a data structure which is also known as binary hash tree.
- This data structure summarizes and verifies the integrity of large sets of data.
- In this data structure, each non-leaf node is a hash of its child nodes and all the leaf nodes are at the same depth.
- In Merkle tree, inputs are first placed at the leaves.
- Both the child nodes are hashed together to generate a value for parent node.
- This procedure is repeated till the single hash value which is Merkle root is generated.
- Following figure shows Merkle tree :-



- The data is always stored in the memory at particular address and pointer stores this memory address of data.
- In hash pointer, pointer to data and cryptographic hash of data is stored.
- Therefore, hash pointer ~~stores this~~ points to data and also helpful in verification of this data.



- Usage of Merkle tree in Blockchain :

1) Data Integrity

⇒ The Merkle root is single, unique fingerprint of all transactions in a block.
- Changing any transaction would change the corresponding leaf hash, which would propagate up the tree, changing the Merkle tree.
- This ensures that any tampering with the transactions can be detected quickly.

2) Scalability

⇒ Merkle tree allows for scalable verification. Instead of verifying each transaction individually, nodes can verify the Merkle root, which proves the integrity of all transactions in the block.

Q.4 List and Explain :

a) Components of blockchain

⇒

- The components that are used in the Blockchain technology :-

1) Node
2) Ledger or Database
3) Wallet
4) Nonce
5) Hash

1) Node

⇒ Any device such as computer, mobile device that is connected to the Blockchain network is called as node.

- These interconnected nodes have copy of blockchain ledger.
- It is of two types :-

a) Full Node : It maintains full copy of all the transactions. It has the capacity to accept, validate and reject the transactions.

b) Partial Node : It is also called as light node because it doesn't maintain the whole copy of blockchain ledger. It maintains only the hash value of the transaction.

2) Ledger / Database

⇒ Ledger is a digital database of transactions or information which can not be altered by anyone. It means they are immutable.

- There are three types of ledger :-

a) Public : It is open and transparent to all. Anyone in the blockchain network can read or write something.

b) Distributed : In this ledger, all nodes have a local copy of database. Here, a group of nodes collectively execute the job.

c) Decentralized : In this ledger, no one node or group of nodes has a central control. Every node participate in the execution of the job.

3) Wallet

⇒ It is a digital wallet that allows users to store their cryptocurrency. Every node in the blockchain network has a wallet.

- Privacy of wallet in a blockchain network is maintained using public and private network.
- In a wallet, there is no need of currency conversion as the currency in the wallet is universally acceptable.

4) Nonce

⇒ Nonce is related to cryptographic communication. It is an unique random number generated and used only once in communication.

- It is added in the identifier of transaction to make it unique so that duplicate transactions can be evoided.
- Nonce is a key to create block in blockchain database.

5) Hash

⇒ The data is mapped to a fixed size using hashing. It plays important role in cryptography.

- Properties of Hash :

i) Collision resistant
ii) Hiding
iii) Puzzle friendliness.

b) Types of blockchain

⇒

- There are four types of blockchain :-

1) Public blockchain
2) Private blockchain
3) Hybrid blockchain
4) Consortium blockchain

1) Public Blockchain

⇒ This is also known as public permissionless blockchain. Bitcoin is public permissionless Blockchain.

- This blockchain is fully decentralized distributed network.
- In this blockchain, everyone is anonymous. It is not necessary to use your real name, or real identity.
- Anyone can access the copy of the ledger and can read, write, participate within the blockchain.
- There is no need to obtain permission for accessing or initiating the transaction
- For. e.g., Bitcoin, Litecoin, Ethereum, etc.

2) Private Blockchain:

⇒ This is also known as private permissionless permissioned blockchain.

- Blockchain network is only open to the participants having access permission.

- In blockchain, permissions are controlled by few nodes in the organization.
- Hence, private blockchain is not fully decentralized, but it is more centralized.
- For e.g., R3 (banks), EWF (Energy), B3i (Insurance), etc.

3) Hybrid

⇒ This type of blockchain is mostly used in government organizations and regulated enterprises.
- It includes best practices and benefits of both private and public Blockchains.
- The hashed data block created in private network is then shared in public network without compromising the privacy of data.

4) Consortium

⇒ It is a creative approach that solves the needs of the organization. This blockchain validates the transaction and also initiates or receives transaction.
- This is also known as federated blockchain which is hybrid between the public and private blockchain.
- It is also permissioned Blockchain having security features just like the features in the public blockchain but it maintains fair amount of control over the network.

Q.5 Explain consensus protocol.

⇒
- A consensus mechanism is used in the blockchain for the agreement between nodes in the network about the valid transactions so that they can be updated on the ledger.
- This mechanism is fault-tolerant and ensures security in blockchain.
- Many consensus algorithms exist those functions based on different principles.
- Some of them are:

1) Proof-of-Work (PoW)

⇒ This consensus algorithm is used to select a miner for the next block generation.
- Bitcoin uses this PoW consensus algorithm.
- The central idea behind this algorithm is to solve a complex mathematical puzzle and easily give out a solution.

2) Proof of Elapsed Time (PoET)

⇒ This algorithm mainly used in permissioned blockchain networks to determine who creates the next block.
- This algorithm follows a fair lottery system which ensures equal chance for each participant to be winner.

3) Proof-of-Stake (PoS)

⇒ This is the most common alternative to PoW.

- In this algorithm, owners of the cryptocurrency can stake their coins, which assigns them right to validate new blocks of transaction and add them to the blockchain.
- If more stake one has in validating nodes, then chances of corrupting the node reduces.
- Ethereum has shifted from PoW to PoS consensus.

4) Delegated Proof of Stake (DPoS)

⇒ This type of consensus mechanism depends on the basis of the delegation of votes.

- The users delegate their votes to other users.
- Whichever user, then mines the block will distribute the rewards to the user who delegated to that particular vote.

Q.6 What do you mean by smart contract? List and explain characterstics of a smart contract.

⇒
- The smart contract is the software program implementing a set of rules or conditions and operates on the top of a blockchain.
- These set of rules are used by different parties to that smart contract to transfer the digital assets between them.
- Real time tracking of performance and transactions is carried out by smart contracts. It also contribute in saving the overall cost.
- Following are some of the important characterstics of the smart contracts :-

1) Independence
2) Cost Effective
3) Trustfulness
4) Recovery and Backup
5) Protection
6) Agility
7) Truth based accuracy.

1) Independence

⇒ Smart contracts can verify and execute itself and are tamper resistance.
- As it is developed by individual node, authorization is not dependent on an

agent, notary or third parties. Hence, tampering by any intermediator not possible.
- They automate legal transactions that defines rules and conditions and also offers high security.

2) Cost effectiveness
⇒ There is no need to have trusted intermediatory in any sell or purchase deal and hence having lower transactions costs.

3) Trustfulness
⇒ As transactions are encoded and encrypted on a distributed replicated ledger, No one can announce that it has been lost.

4) Recovery and Backup
⇒ It is possible as same copies are available on multiple nodes.

5) Protection
⇒ Addresses are encrypted and hence contracts are well-preserved and prevented from any malicious activities.

6) Agility
⇒ All the business-related activities to be carried out are done/executed automatically

7) Truth based Accuracy

⇒ There is no intervention of human and smart contracts gets executed automatically Hence, operation is error free.

Q. 7 Explain the structure of smart contract solidity program.

⇒

- A contract is like a class which contains state variables, functions, function modifiers, events, structures, and enums.

1) State variable

⇒ State variables are stored in contract storage. These can be set as constants.

- Expensive to use, they cost gas.
- For e.g.,

```
pragma solidity ^0.8.2;
contract myContract {

    uint myData;
}
```

2) Function

⇒ Functions are like a few lines of code that can be perform something within the contract.

- For e.g.,

```
contract Customer {
    string id = 34;

    function getId() public constant
        returns (string) {
            return id;
        }
}
```

3) Function modifiers

⇒ Function modifier are similar to a function that checks the validation rules prior to executing the main function.

- For e.g.,

```
contract Purchase {

    address public seller;

    modifier onlySeller() {
        require (msg.sender == seller);
        _;
    }

    function abort() public onlySeller {
        // ...
    }
}
```

4) struct types

⇒ struct is custom defined data types that consists of several variables declared in it.

- For e.g.;

```
pragma solidity ^0.8.2;

contract Node {
    struct caller {
        uint id;
        bool isExist;
        address delegate;
        uint age;
    }
}
```

5) Events

⇒ Events are functions that are bubbled up after compilation of function execution if event code written there, it provide EVM logging functionality.

- For e.g.;

```
pragma solidity ^0.8.2;

contract Auction {
    event HighestBid (address, user, unit amt);

    function bid() public payable {
        emit HighestBid (msg.sender, msg.value);
    }
}
```

6) Enum types

⇒ Enum is used to create a custom types and it has fixed the set of values.

- For eg.,

```
pragrama solidity ^0.8.2;

contract Shopping{

    enum KIT { cricket, hockey, tabletennis}
}
```

Q.8 List and explain types of a function visibility in solidity.

⇒

- In solidity, functions have different visibility types that determines how and where they can be accessed and called.
- The types of function visibility in solidity:-

1) Public
2) External
3) Internal
4) Private.

1) Public

⇒ A public function can be called both internally and externally.
- For a public state variable, an automatic getter function is generated.
- For e.g.)

```
function set(uint x) public {
    storeData = x;
}
```

- In this example, the 'set' function is public, meaning it can be called by any external account or contract, as well as from within the contract itself.

2) External

⇒ An external function can only be called from outside the contract.

- It cannot be called internally from within the same contract unless using 'this.functionName()'.
- For e.g.,

```
function updateData(uint x) external {
    storeData = x;
}
```

3) Internal

⇒ An internal function can only be called from within the same contract or from derived contract. It cannot be called externally.

- For e.g.,

```
function calculate(uint x, uint y) internal {
    return x+y;
}
```

4) Private

⇒ A private function can be called from within the same contract.

- It cannot be called by derived contracts or from outside the contract.

- For e.g.;

```
function increment(uint x) private pure
returns (uint) {

    return x+1;
}
```

Q. 9 Write a program in solidity to implement

a) multi-level inheritance

⇒ Multi-level inheritance involves a contract inheriting from another contract, which in turns inherits from yet another contract. This forms a chain of inheritance.

```
pragma solidity ^0.8.2;

contract BaseContract {
    uint public Bval;

    constructor(uint _Bval) {
        Bval = _Bval;
    }

    function setValue(uint _Bval) public {
        Bval = _Bval;
    }
```

```
    function getvalue() public view returns
    (uint) {
        return Bval;
    }
}

contract DerivedContract is BaseContract{

    uint public Dval;

    constructor (uint _Bval, uint _Dval)
    BaseContract(_Bval) {
        Dval = _Dval;
    }

    function setvalue (uint _Dval) public{
        Dval = _Dval;
    }

    function getvalue() ~~uint~~ public view
    returns (uint){

        return Dval;
    }
}
```

```
contract FinalContract is DerivedContract{

    uint public Fval;

    constructor(uint _Bval, uint _Dval,
        uint _Fval){
        Fval = _Fval;
    }

    function setValue( uint _Fval) public {
        Fval = _Fval;
    }

    function getValue() public view
        returns(uint){
        return Fval;
    }
}
```

b) Check whether number is Prime or Not

⇒

```
pragma solidity ^0.8.2;

contract Prime {

    function isPrime(uint num) public pure
    returns (bool) {

        if (num <= 1) {
            return false;
        }

        for (uint i=2; i*i <= num; i++) {
            if (num % i == 0) {
                return false;
            }
        }

        return true;
    }
}
```

Q.10 What is the fallback function in solidity? Explain.

=>

- 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 fallback function :-

1) Declare with fallback() and have no arguments.
2) If it is not marked payable, the contract will throw an exception if it receives plain ether without data.
3) Cannot return anything.
4) Can be defined once per contract.
5) It is mandatory to mark it external.

- For e.g.)

~~program~~

```
pragma solidity ^0.8.21

contract temp {

    event LogFallbackCalled (address sender,
    uint amt, bytes data);

    fallback() external payable() {

        emit LogFallbackCalled (msg.sender,
        msg.value, msg.data);
    }
}
```