Assignment 3

<u>Title</u>:-
Write a smart contract on a test network, for Bank account of a customer for following operations:
1. Deposit money.
2. Withdraw money.
3. Show balance.

<u>Objectives</u>:-

1. Understand the working of blockchain.
2. Learn about smart contract.

<u>Requirements</u>:-

- Any browser
- Remix IDE.
- Metamask wallet.

<u>Theory</u>:-

What is a smart contract?

Smart contracts are immutable programs stored on a blockchain. They automate the execution of transactions based on predetermined conditions being met, and they are widely used to execute agreements in a decentralized manner without middlemen.
Smart contracts have particular outcomes,

which are governed by immutable code, so the participants in the contract can be confident in the contract execution. No third-party involvement, no time lost- agreements are executed immediately when the conditions are met.

Smart contracts can be deployed on the blockchain for use. Ethereum supports smart contracts written in the solidity programming language.

Every smart contract is owned by an address called as owner. A smart contract can know its owner's address using sender property and its available balance using a special built-in object called msg.

Banking contract:

The contract will all deposits from any account, and can be trusted to allow withdrawals only by accounts that have sufficient funds to cover the requested withdrawal.

```
address owner;   // current owner
function TipJar () public {
    owner = msg.sender;
}
function withdraw () public {
    require (owner == msg.sender);
```

```
        msg.sender.transfer(address(this).balance);
    }
    function deposit(uint256 amount) payable {
        require(msg.value == amount);
    }
    function getBalance() public view returns (uint25
    {
        return address(this).balance;
    }
}
```

Maintaining individual account balances:
To generalize this contract to keep track of ether deposits based on the account address of the depositor, and then only allow that same account to make withdrawals of that ether.

```
    mapping (address => uint256) public balanceOf;
        // balances, indexed by addresses.
    function deposit(uint256 amount) public payable
    {
        require(msg.value == amount);
        balanceOf[msg.sender] += amount;
    }
}
```

Rajdhani
DATE / /

With drawals and Account Balances:

```
mapping(address => uint 256) public balanceOf;
function deposit (uint 256 amount)
public payable {
      require (msg.value == amount);
      balance of [msg.sender] += amount;
  }
  function withdraw (uint 256 amount)
  public {
        require (amount <= balance of [msg.sender]);
        balance of [msg. sender] -= amount;
        msg. sender. transfer (amount);
   }
}
```

So, solidity supports a key value datatype called mapping The default value associated with a missing key is 0.
A mapping (address => uint 256) enables straight forward accounting of per-account ether balances.

Conclusion :-
Hence, we have studied about smart contracts on a test network, for bank account of a customer.