

| **Title:**  Development of smart contract for any simple application – I |
| --- |

**Objective:** Learn Solidity language ;use algorithms & smart contract and develop an application in Blockchain.

**Expected Outcome of Experiment:**

| **CO** | **Outcome** |
| --- | --- |
| CO2 | Learn Solidity language & Multiple Technology-based developments. |
| CO3 | Apply the algorithm and techniques used in Blockchain. |

**Books/ Journals/ Websites referred:**

* Remix IDE
* https://en.wikipedia.org/wiki/Smart\_contract
* https://soliditylang.org/

**Abstract**:-

The concept of smart contracts is primarily based on the idea of blockchain technology. A blockchain is a decentralized network of a growing list of records (blocks) that are linked through cryptography. A blockchain network does not include a single central point like a conventional database. The data that is stored in the blockchain is shared between all the computers that comprise the network. Therefore, the network is less exposed to possible failures or attacks.

In addition, in a blockchain, a record in one computer cannot be altered without changing the same record on other machines in the network. Transactions executed through a blockchain are grouped in blocks that are linked in a chain. A new block is created only when the previous block is completed. The blocks come in a linear chronological order, and each block contains a cryptographic hash of the previous block.

**Related Theory: -**

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.It is deployed to blockchains that render transactions traceable, transparent, and irreversible. Smart contracts are verified, executed, and enforced by a computer program that runs on a blockchain network.

When both parties involved in the smart contract agree to its terms, the program will automatically execute. This eliminates the need for a third party, as the contract is verified and enforced by the blockchain network. Because smart contracts are executed by code rather than people, they remove the possibility of human error and can automate many tasks that would traditionally require human interaction.

One of the best things about the blockchain is that, because it is a decentralized system that exists between all permitted parties, there’s no need to pay intermediaries (middlemen) and it saves you time and conflict. This makes more smart contracts being executed on different blockchain networks including Ethereum, Solana, Tezos, Hyperledger, etc.

**Implementation Details:**

**1. Enlist all the Steps followed and various options explored**

* Initialized the contract
* Declared the variables payer,payee,amount,thirdParty
* Defined & intialized the constructor with payable
* Defined deposit(), release() & balanceOf() using required() and msg.sender
* On IDE we set the payer, payee, amount and deploy from thridParty account and pay Gas.
* Till the contract balance amount is not deposit fully; it is not released by thirdParty
* Following the all conditions correctly we transfer the contract balance amount from A to B via escrow C.

**2. Explain your program logic, classes and methods used.**

* constructor ( ) - It takes three arguments to initialize the payer, payee and amount.
* deposit ( ) -

1. It will take deposit sent by the payer only . If the payer is not the depositer then it prints "Sender must be the payer" .
2. If the deposit is done by the depositer then it will check whether the amount sent by the payer is less than or equal to contract balance or not. If it is not then prints "Cant send more than escrow amount" .

* release()

1. It will check whether the amount is equal to the contract balance or not. If not it prints "cannot release funds before full amount is sent" .
2. If the balance is equal to amount. Then it will check whether the release ( ) by thirdParty or not. If not prints "only thirdParty can release funds" .
3. If release( ) is called by the thirdParty. Then it transfers the amount to the payee.

* balanceOf() - returns the balance of the contract

**3. Explain the Importance of the approach followed by you**

This approach helps us to transfer the contract balance amount from A to B via escrow C. It helps to maintain smooth functioning of amount transfer by establishing trust between the parties.

It helps to check that only payer is paying & payee is receiving the amount and only release rights to thirdParty. It helps to check if contract balance is sufficient and as per decided amount. This makes Escrow Smart Contract easy in Solidity and using Remix IDE.

**Code:**

// SPDX-License-Identifier: GPL-3.0

pragma solidity >=0.7.0 <0.9.0;

contract escrow {

address public payer;

address public thirdParty;

address public payee;

uint256 public amount;

constructor(

address \_payer,

address \_payee,

uint256 \_amount

) payable {

amount = \_amount \* (10\*\*18);

payee = \_payee;

payer = \_payer;

thirdParty = msg.sender;

}

function deposit() public payable {

require(payer == msg.sender, "Sender must be the payer");

require(

amount >= address(this).balance,

"Cant send more than escrow amount"

);

}

function release() public payable {

require(

amount == balanceOf(),

"Cannot release the funds before full amount is sent"

);

require(msg.sender == thirdParty, "only thirdParty can release funds");

payable(payee).transfer(balanceOf());

}

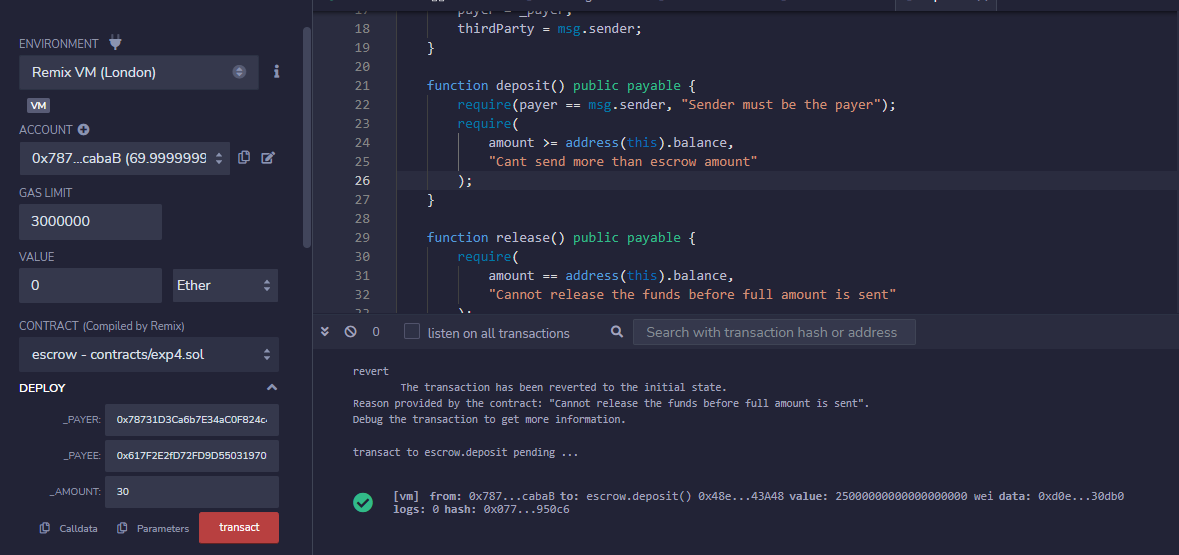
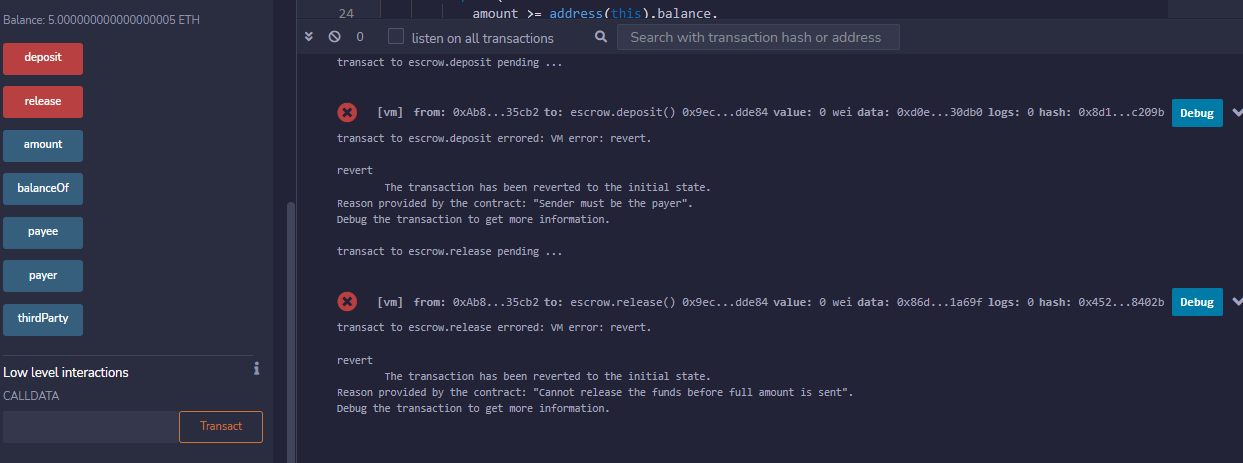
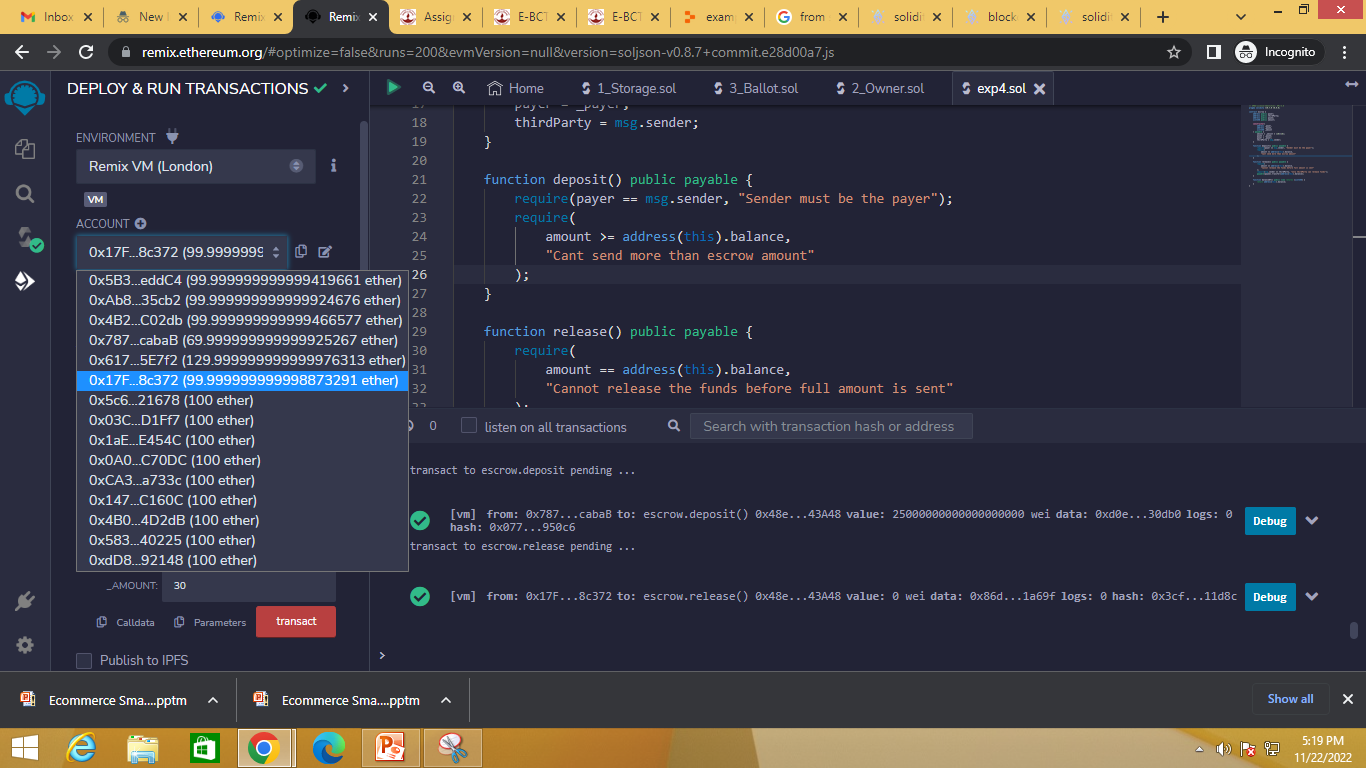
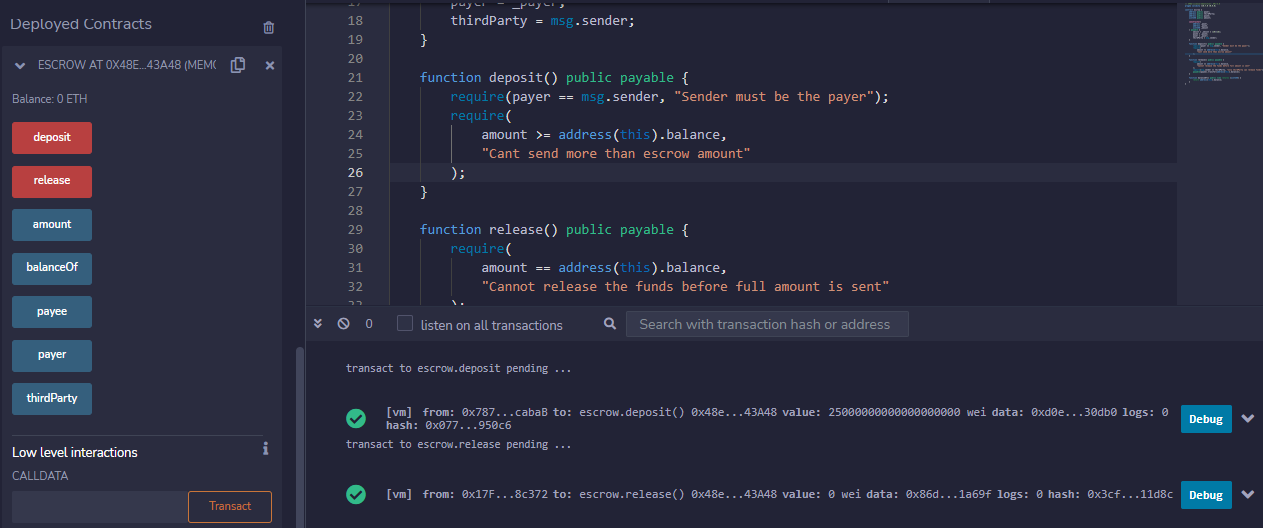
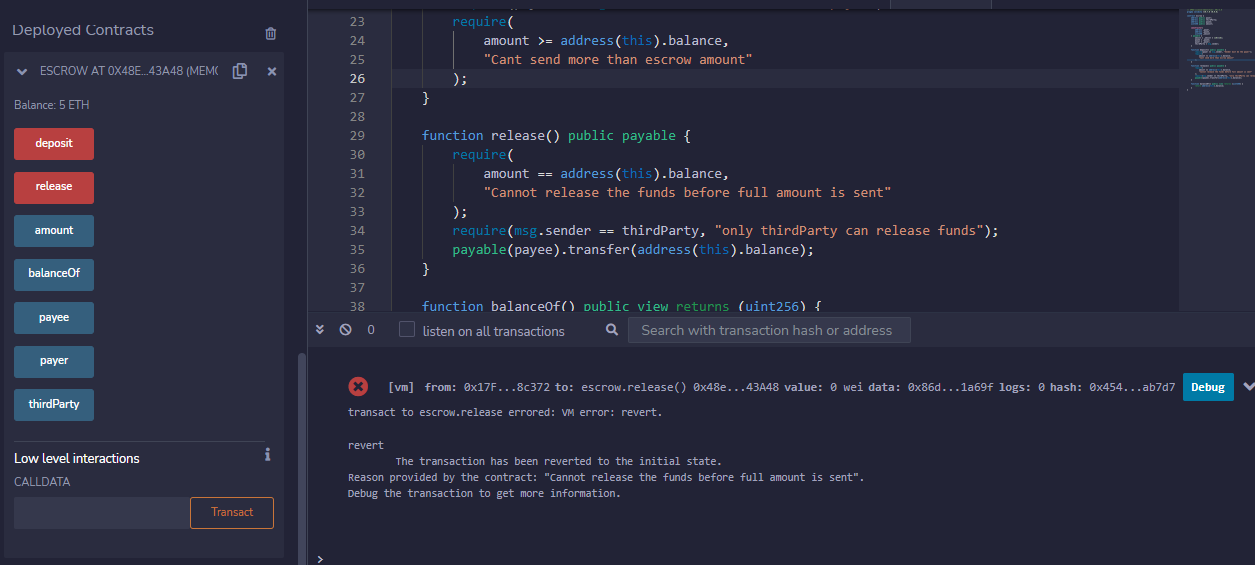
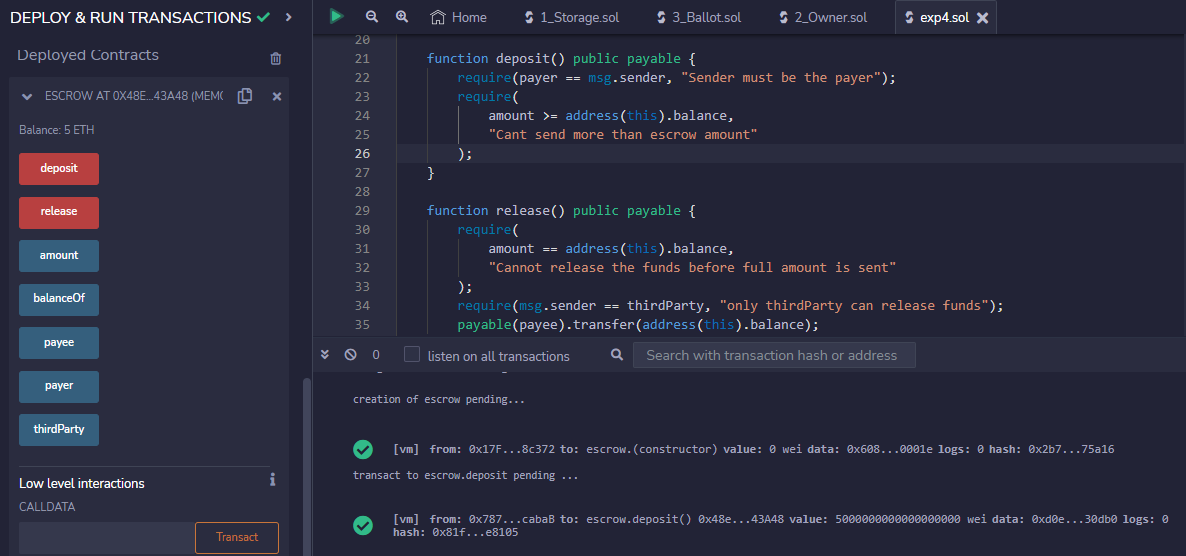
function balanceOf() public view returns (uint256) {

return address(this).balance;

}

}

**Output:**



**Conclusion:-**

We understood and implemented the smart contract & algorithms used in Blockchain in Solidy using Remix IDE.