BT

Bank Program

pragma solidity ^0.8.4;

contract Bank {

mapping(address => uint) balance;

address owner;

constructor() {

owner = msg.sender;

}

function addBalance(uint \_toAdd) public returns(uint) {

require(msg.sender == owner,"not the owner");

balance[msg.sender] += \_toAdd;

return balance[msg.sender];

}

function getBalance() public view returns(uint) {

return balance[msg.sender];

}

function Withdraw(uint \_tosub) public returns(uint) {

require(msg.sender == owner,"not the owner");

balance[msg.sender] -= \_tosub;

return balance[msg.sender];

}

}

Student Data

//SPDX-License-Identifier: MIT

pragma solidity ^0.8.7;

contract MarksManagmtSys

{

struct Student //struct user defined data type

{

int ID;

string fName;

string lName;

int marks;

}

address owner; //owner address return

int public stdCount = 0;

mapping(int => Student) public stdRecords; //maps student id at position of int

modifier onlyOwner

{

require(owner == msg.sender);

\_;

}

constructor()

{

owner=msg.sender;

}

function addNewRecords(int \_ID, //fun to add new record

string memory \_fName,

string memory \_lName,

int \_marks) public onlyOwner

{

stdCount = stdCount + 1; //new stu will be added

stdRecords[stdCount] = Student(\_ID, \_fName,

\_lName, \_marks);

}

function bonusMarks(int \_bonus) public onlyOwner //fun to add bonus marks

{

stdRecords[stdCount].marks =

stdRecords[stdCount].marks + \_bonus;

}

}

////////////////////////////////////////////////////////////////////////////////////

//SPDX-License-Identifier:MIT

pragma solidity ^0.8.0; //version

contract Bank{

address private owner; //private data we cant see directly (address=datatype, owner is name of var ehich is set to private)

constructor() public {

owner=msg.sender; //inbuild fun(msg.sender)

}

mapping(address=>uint256) public Balance; //mapping the balance of owner

function Deposit() public payable { //fun to deposit money/ether

Balance[msg.sender]+=msg.value; //whatever we add bal will go to sender fun

}

function getOwner() public view returns(address){ //returns address of owner

return owner;

}

function withDraw(uint256 \_amount2Withdraw) public { //withdraw money

Balance[msg.sender]-=\_amount2Withdraw;

(bool sent,)=msg.sender.call("sent");

require(sent,"Not able to send!"); //if(require=if in solidity

}

function ShowBalance(address myUser) public view returns(uint256){ //fun to show bal

return Balance[myUser];

}

}