**4.Student marks**

/ Solidity program to implement

// the above approach

pragma solidity >= 0.7.0<0.8.0;

// Build the Contract

contract MarksManagmtSys

{

// Create a structure for

// student details

struct Student

{

int ID;

string fName;

string lName;

int marks;

}

address owner;

int public stdCount = 0;

mapping(int => Student) public stdRecords;

modifier onlyOwner

{

require(owner == msg.sender);

\_;

}

constructor()

{

owner=msg.sender;

}

// Create a function to add

// the new records

function addNewRecords(int \_ID,

string memory \_fName,

string memory \_lName,

int \_marks) public onlyOwner

{

// Increase the count by 1

stdCount = stdCount + 1;

// Fetch the student details

// with the help of stdCount

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

\_lName, \_marks);

}

// Create a function to add bonus marks

function bonusMarks(int \_bonus) public onlyOwner

{

stdRecords[stdCount].marks =

stdRecords[stdCount].marks + \_bonus;

    }

}

**3. Bank code**

// SPDX-License-Identifier: MIT

pragma solidity >=0.7.0 <0.9.0;

// Contract for a simple bank account

contract SimpleBank {

    address public owner;

    mapping(address => uint) private balances;

    // Modifier to allow only the owner to perform certain actions

    modifier onlyOwner() {

        require(msg.sender == owner, "Only the owner can perform this action.");

        \_;

    }

    // Constructor to set the owner of the contract

    constructor() {

        owner = msg.sender;

    }

    // Function to deposit money into the account

    function deposit() public payable {

        require(msg.value > 0, "Deposit amount must be greater than zero.");

        balances[msg.sender] += msg.value;

    }

    // Function to withdraw money from the account

    function withdraw(uint amount) public {

        require(amount > 0, "Withdraw amount must be greater than zero.");

        require(balances[msg.sender] >= amount, "Insufficient balance.");

        balances[msg.sender] -= amount;

        payable(msg.sender).transfer(amount);

    }

    // Function to show the balance of the account

    function getBalance() public view returns (uint) {

        return balances[msg.sender];

    }

}