**BT**

Practical 3

// SPDX-License-Identifier: MIT

pragma solidity ^0.8.1;

contract SendMoneyExample{

uint public balanceReceived;

function receiveMoney() public payable {

balanceReceived += msg.value;

}

function getBalance() public view returns(uint){

return address(this).balance;

}

function withdrawMoneyTo(address payable \_to) public{

\_to.transfer(getBalance());

}

}

Practical 4

// SPDX-License-Identifier: MIT

pragma solidity ^0.8.1;

contract MarksManagementSys{

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;

}

function addNewRecords(int \_ID,

string memory \_fName,

string memory \_lName,

int \_Marks) public onlyowner

{

stdCount = stdCount + 1;

stdRecords[stdCount]=student(\_ID,\_fName,\_lName,\_Marks);

}

function bounsMarks(int \_bouns) public onlyowner

{

stdRecords[stdCount].Marks =

stdRecords[stdCount].Marks + \_bouns;

}

receive() external payable {

revert("Receive function: This contract does not accept Ether.");

}

}