Source Code of Assignment

1. Create multiple accounts in Metamask and perform the balance transfer between the various accounts

pragma solidity ^ 0.6.0;

contract mycontract {

    uint myVar;

    function seVar(uint newVar) public {

        myVar = newVar;

    }

    function getVar() public view returns (uint) {

        return myVar;

    }

 }

2. 2. Write a solidity program to set variable and get variable

pragma solidity ^ 0.6.0;

contract mycontract {

    uint myVar;

    function seVar(uint newVar) public {

        myVar = newVar;

    }

    function getVar() public view returns (uint) {

        return myVar;

    }

 }

3. 3. Write a solidity program to perform push and pop operations on dynamic array

pragma solidity ^0.6.0;

contract testArray{

uint[] public dynamicArray;

function setdynamicArray(uint value) public {

dynamicArray.push(value);

}

function removeValue() public{

dynamicArray.pop();

}

function getlength() public view returns(uint length){

return dynamicArray.length;

}

}

4. Write a solidity program to set address with a mapping variable

pragma solidity ^0.6.0;

contract testMapping{

    mapping (uint => bool) public myMapping;

    mapping (address => bool) public myAddress;

    function setValue(uint index) public{

        myMapping[index] = true;

    }

    function setAddress(address add) public {

        myAddress[msg.sender]=true;

    }

}

5. Write a solidity program to get the factorial of a number

pragma solidity ^0.6.0;

contract Factorial {

  function fact(uint x) public view returns (uint y) {

    if (x == 0) {

      return 1;

    }

    else {

      return x= x\*fact(x-1);

    }

  }

}

6. Write a solidity program to store information of a student(Name, Roll.No, Institute, Age) using structure

pragma solidity 0.6.0;

contract Studentdetails{

    Student[] public student;

    uint256 public studentCount;

    struct Student {

        string \_firstName;

        string \_lastName;

        uint256 id;

        uint256 age;

        string \_InstituteName;

    }

    function addPerson(string memory \_firstName, string memory \_lastName, uint256 id, uint256 age, string memory \_InstituteName) public {

        student.push(Student(\_firstName, \_lastName, id, age, \_InstituteName));

    }

}

7. Write a smart contract using a solidity program to perform balance transfer from contract to other accounts.

pragma solidity ^0.6.0;

contract  balanceTransfer{

    function receivemoney() public payable {

        //totalbalance += msg.value;

    }

    function getbalance() public view returns(uint){

        return address(this).balance;

    }

    function transferMoney(address payable toAccount, uint amount) public{

        toAccount.transfer(amount);

    }

}

8. Write a smart contract using a solidity program to perform balance transfer with mapping and make sure only the owner can transfer the balance from contract to other contract.

pragma solidity ^0.6.0;

contract BalanceTransfer{

    address owner;

    mapping(address => uint) public totalBalance;

    constructor() public {

        owner = msg.sender;

    }

    function receiveBalance() public payable {

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

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

    }

    function getBalance() public view returns(uint){

        return address(this).balance;

    }

    function transferBalance(address payable toAccount, uint amount) public {

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

        require(totalBalance[msg.sender] >=amount, "Insufficient Balance");

        toAccount.transfer(amount);

        totalBalance[msg.sender] -= amount;

    }

}

9. Write a solidity program to perform the exception handling and describe the details with screenshots