**Hands on 4 and 5**

**Task 1: Working on solidity on Remix IDE- topics to be covered (23/03/2023-2 hours)**

* 1. Mapping
  2. Conversions
  3. Ether units
  4. Functions
  5. Functions modifier
  6. View, pure, fallback functions
  7. Function overloading
  8. Built in functions – mathematical and cryptography
  9. Withdrawn pattern
  10. Restricted access

**Task 2: Understanding the code flow (30/03/2023-2 hours)**

To be done by students. The aim of this task is to make them familiar with what the contract is explaining and how the input is to be given to get the correct output. Attendance will be given only if they submit this successfully in the given time period.

1. Execute the following program and attach screenshots of the output generated for the following test cases:
   1. 4 and 3
   2. 256 and 256
   3. 9 and 12

* State what the given program is doing by explaining the outputs generated in each test case.
* Explain what **Loop1** and **Loop2** does in the given code.
* Replace the statement in **Position 3** with a single line statement that does the same role/task as the statement(/s) given.
* What does **small** being assigned to 0 in the if condition at **Position 4** indicate?
* Make appropriate changes to the **Mystery function** to return the two values without making use of a return statement. Attach a screenshot of the updated function (including the replacement line at Position 3).

pragma solidity ^0.8.7;

contract Lab\_4 {

   constructor() public{

   }

   function getResult(uint var1, uint var2) public pure returns(uint, uint ){

      uint result = var1 + var2;

      return Mystery(result);

   }

   function Mystery(uint \_i) internal pure  returns (uint, uint) {

      if (\_i == 0) {

         return (0,1);

      }

      uint len=8;

      while(\_i > (2\*\*len)-1)  //LOOP 1

      {

          len+=8;

      }

      uint len\_actual = len/8;

       uint j = 2;

      uint small =9999;

      while (j!=\_i){ //LOOP2

        if (\_i%j== 0)

        {

            if(j<small){  // POSITION 3

                small=j;

            }

        }

        j++;

      }

      if (small==9999){  // POSITION 4

          small=0;

      }

      return (\_i<<small,len\_actual); // POSITION 5

   }

}

1. Write the Solidity code for a contract **Contract\_XYZ** that follows the given outline-

**Set\_Method(**) function takes the size of the array (**a1**)  and the multiplication factor(**a\_m**) from the user. The function returns the multiplication table of a\_m till size a1.

Example:

**Input**: 5(**a1**), 15(**a\_m**)

**Output**: 0,15,30,45,60

Attach appropriate code and output screenshots in the document before submitting.

1. Analyse the given codes on REMIX platform Metamask environment variable.

Take appropriate screenshots of the outputs and meta mask wallets after deploying the smart contracts.

**Code 1**

// SPDX-License-Identifier: GPL-3.0

pragma solidity ^0.8.7;

contract C {

   //private state variable

   uint private data;

   //public state variable

   uint public info;

   //constructor

   constructor() public {

      info = 10;

   }

   //private function

   function increment(uint a) private pure returns(uint) { return a + 1; }

   //public function

   function updateData(uint a) public { data = a; }

   function getData() public view returns(uint) { return data; }

   function compute(uint a, uint b) public pure returns (uint) { return a + b; }

}

//External Contract

contract D {

   function readData() public returns(uint) {

      C c = new C();

      c.updateData(7);

      return c.getData();

   }

}

//Derived Contract

contract ReceiveEther {

    constructor() public{

    }

    receive() external payable{}

    address owner2 = msg.sender;

    function Balance() public view returns(uint){

        return owner2.balance;

    }

}

contract SendEther{

    //address owner = msg.sender;

    function sendViaTransfer(address payable \_to) public payable {

        // This function is no longer recommended for sending Ether.

        \_to.transfer(msg.value);

    }

}

**Code 2**

         // SPDX-License-Identifier: GPL-3.0

pragma solidity ^0.6.6;

contract BankContract {

    struct client\_account{

    int client\_id;

    address client\_address;

    uint client\_balance\_in\_ether;

    }

    client\_account[] clients;

    int clientCounter;

    address payable manager;

    mapping(address => uint) public interestDate;

    modifier onlyManager() {

        require(msg.sender == manager, "Only manager can call this!");

        \_;

    }

    modifier onlyClients() {

        bool isclient = false;

        for(uint i=0;i<clients.length;i++){

            if(clients[i].client\_address == msg.sender){

                isclient = true;

                break;

            }

        }

        require(isclient, "Only clients can call this!");

        \_;

    }

    constructor() public{

        clientCounter = 0;

    }

    receive() external payable { }

    function setManager(address managerAddress) public returns(string memory){

        manager = payable(managerAddress);

        return "";

    }

    function joinAsClient() public payable returns(string memory){

        interestDate[msg.sender] = now;

        clients.push(client\_account(clientCounter++, msg.sender, address(msg.sender).balance));

        return "";

    }

    function deposit() public payable onlyClients{

        payable(address(this)).transfer(msg.value);

    }

    function withdraw(uint amount) public payable onlyClients{

        msg.sender.transfer(amount \* 1 ether);

    }

    function sendInterest() public payable onlyManager{

        for(uint i=0;i<clients.length;i++){

            address initialAddress = clients[i].client\_address;

            uint lastInterestDate = interestDate[initialAddress];

            if(now < lastInterestDate + 10 seconds){

                revert("It's just been less than 10 seconds!");

            }

            payable(initialAddress).transfer(1 ether);

            interestDate[initialAddress] = now;

        }

    }

    function getContractBalance() public view returns(uint){

        return address(this).balance;

    }

}