Course: Laboratory Practice III **Course Code: 410246**
Name: Anuj Mutha **Class: BE-4**
Roll No. : 41443 **Batch: R4**

Code:

```
//SPDX-License-Identifier: MIT
pragma solidity ^0.6;

contract banking{
   mapping(address=>uint) public user_account;
   mapping(address=>bool) public user_exists;

   function create_account() public payable returns(string memory)
   {
      require(user_exists[msg.sender]==false,'Account already created');
      if(msg.value==0)
      {
         user_account[msg.sender]=0;
         user_exists[msg.sender]=true;
         return "Account Created";
      }
      require(user_exists[msg.sender]==false,"Account Already Created");
       user_account[msg.sender]=msg.value;
      user_exists[msg.sender]=true;
      return "Account Created";
   }
   function deposit() public payable returns(string memory)
   {
      require(user_exists[msg.sender]==true,"Account not Created");
      require(msg.value>0,"Value for deposit is zero");
      user_account[msg.sender]=user_account[msg.sender]+msg.value;
      return "Deposited Successfully";
   }
   function withdraw(uint amount) public payable returns(string memory)
   {
      require(user_account[msg.sender]>amount,"Insufficient balance");
      require(user_exists[msg.sender]==true,"Account not created");
      require(amount>0,"Amount should be more than zero");
      user_account[msg.sender]=user_account[msg.sender]-amount;
      msg.sender.transfer(amount);
      return "Withdraw Successful";
   }

   function transfer(address payable userAddress,uint amount) public returns(string memory)
```

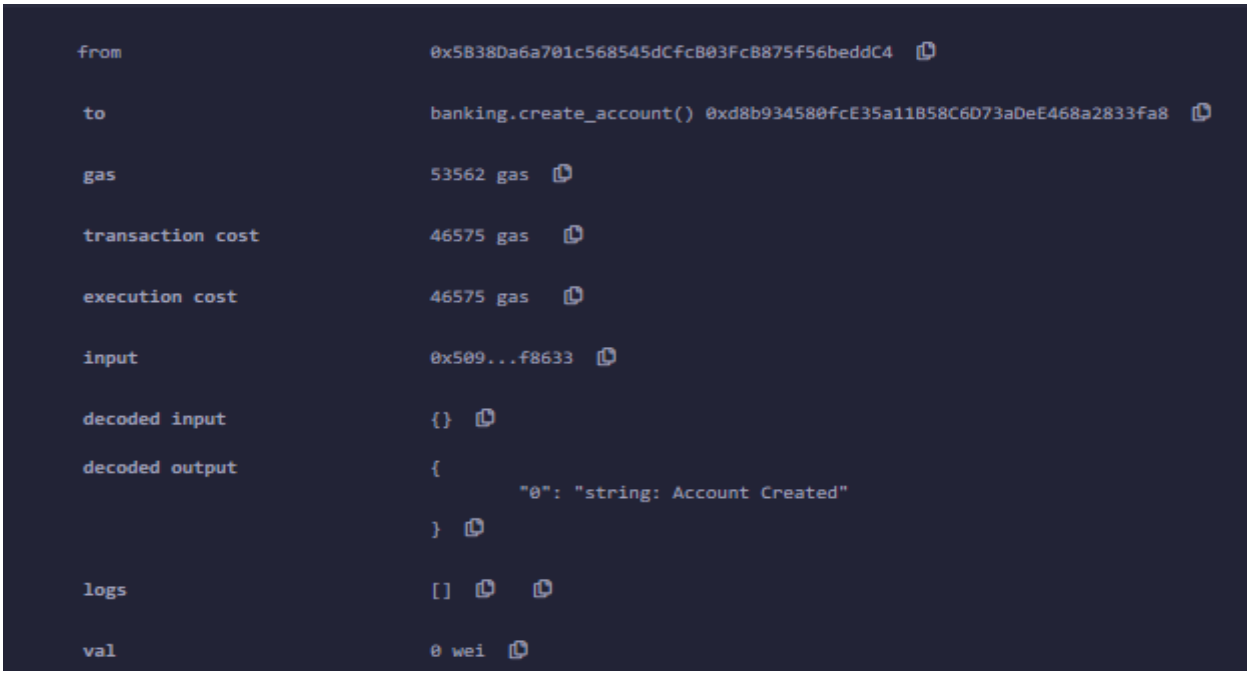
```
    {
        require(user_account[msg.sender]>amount,"Insufficient balance in bank account");
        require(user_exists[msg.sender]==true, "Account is not created");
        require(user_exists[userAddress]==true,"Transefer account does not exist");
        require(amount>0,"Amount should be more than zero");
        user_account[msg.sender]=user_account[msg.sender]-amount;
        user_account[userAddress]=user_account[userAddress]+amount;
        return "Transfer successful";
    }

    function send_amt(address payable toAddress,uint56 amount) public payable returns(string memory)
    {
        require(user_account[msg.sender]>amount,"Insufficient balance in bank account");
        require(user_exists[msg.sender]==true,"Account is not created");
        require(amount>0,"Amoun should be more than zero");
        user_account[msg.sender]=user_account[msg.sender]-amount;
        toAddress.transfer(amount);
        return "Transfer success";
    }

    function user_balance() public view returns(uint)
    {
        return user_account[msg.sender];
    }
    function account_exist() public view returns(bool)
    {
        return user_exists[msg.sender];
    }

}
```

OUTPUT:


from 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
to banking.create_account() 0xd8b934580fcE35a11B58C6D73aDeE468a2833fa8
gas 53562 gas
transaction cost 46575 gas
execution cost 46575 gas
input 0x509...f8633
decoded input {}
decoded output {
"0": "string: Account Created"
}
logs []
val 0 wei

```
CALL [call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: banking.account_exist() data: 0xcde...6e57b

from                    0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
to                      banking.account_exist() 0xd9145CCE52D386f254917e481eB44e9943F39138
execution cost          23552 gas (Cost only applies when called by a contract)
input                   0xcde...6e57b
decoded input           {}
decoded output          {
                            "0": "bool: true"
                        }
logs                    []
```

call to banking.user_balance

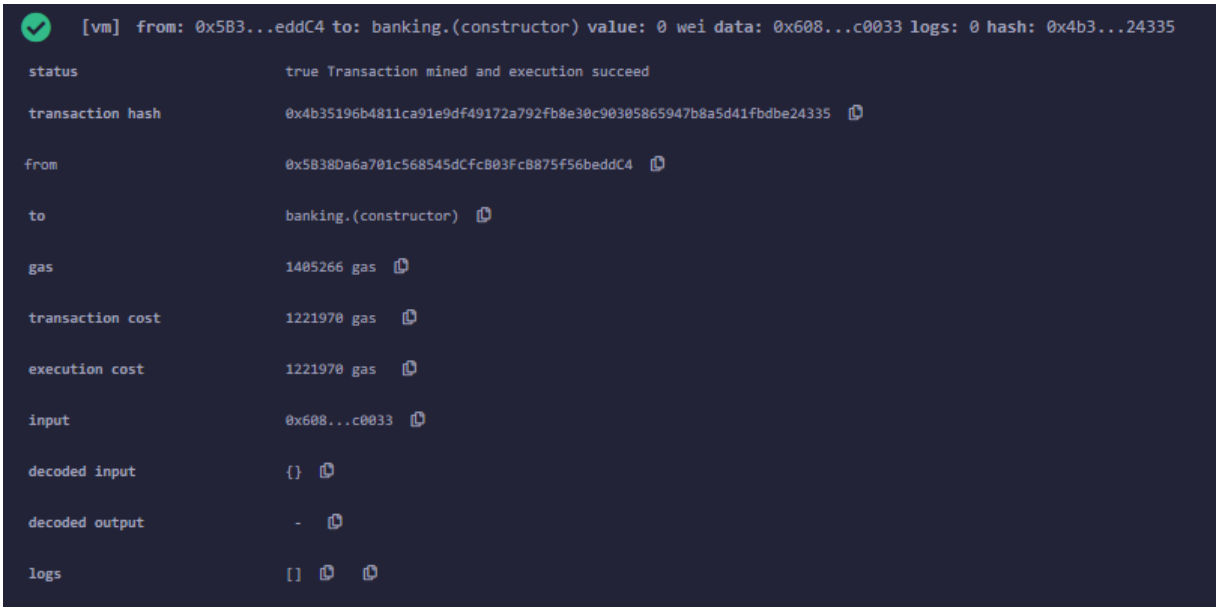
```
CALL [call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: banking.user_balance() data: 0xd3d...a43b3

from                    0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
to                      banking.user_balance() 0xd9145CCE52D386f254917e481eB44e9943F39138
execution cost          23554 gas (Cost only applies when called by a contract)
input                   0xd3d...a43b3
decoded input           {}
decoded output          {
                            "0": "uint256: 0"
                        }
```

call to banking.user_exists

```
CALL [call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: banking.user_exists(address) data: 0x15b...c9f2c

from                    0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
to                      banking.user_exists(address) 0xd8b934580fcE35a11B58C6D73aDeE468a2833fa8
execution cost          23933 gas (Cost only applies when called by a contract)
input                   0x15b...c9f2c
decoded input           {
                            "address ": "0x24AfB438f4AD7c77fd133D2266C93c6B4A3C9F2c"
                        }
decoded output          {
                            "0": "bool: false"
                        }
logs                    []
```



Conclusion:

I studied about smart contract and how to write and execute it using remix ide.