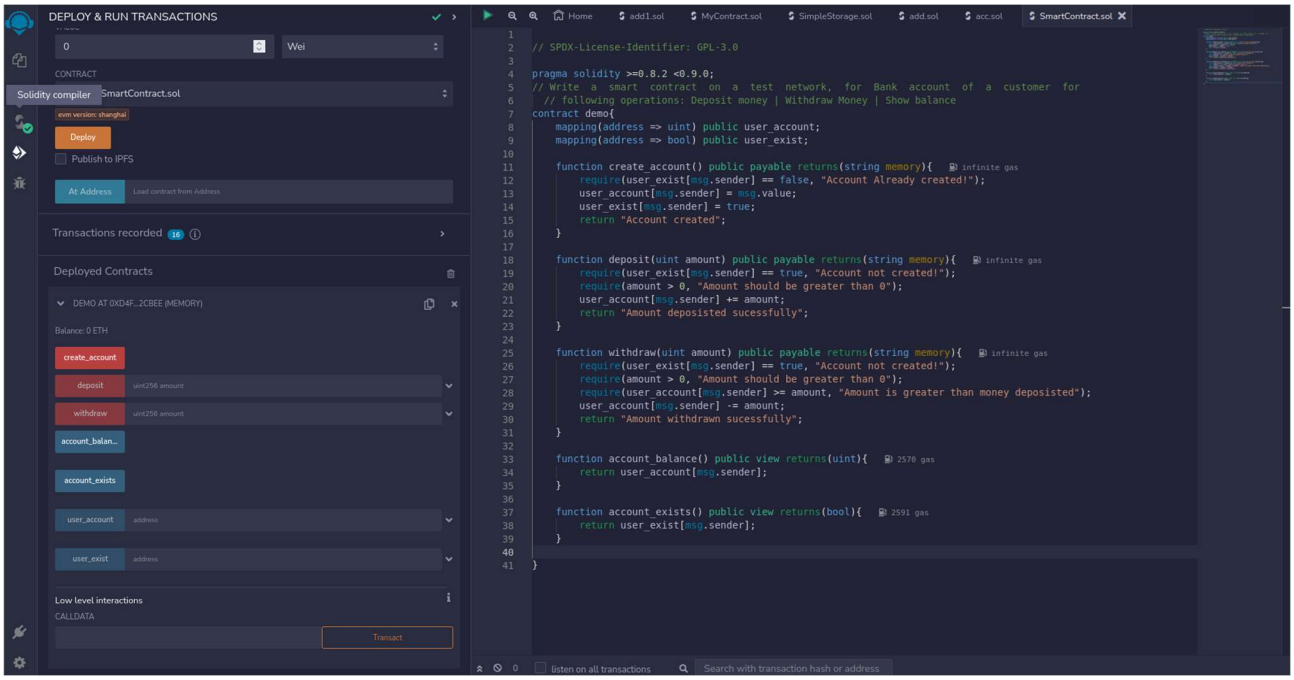
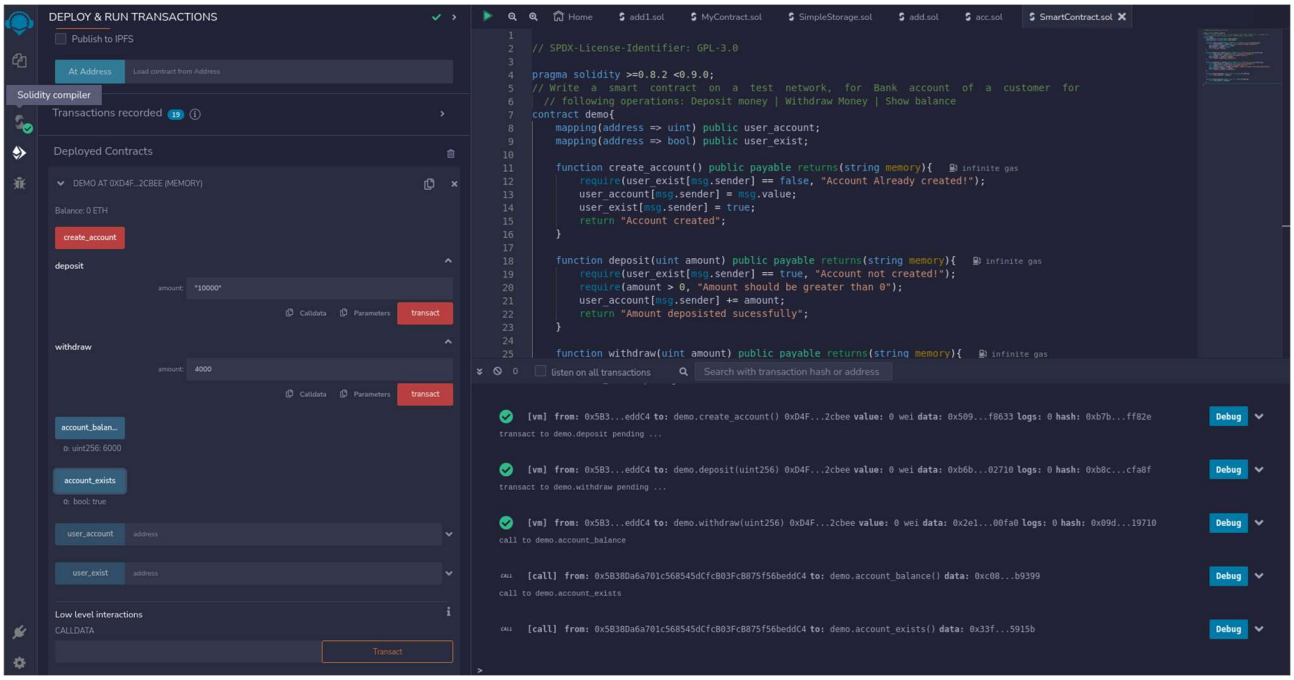
BC220 BT ASSIGNMENT: - 3

Program –

```
// SPDX-License-Identifier: GPL-3.0

pragma solidity >=0.8.2 <0.9.0;
// Write a smart contract on a test network, for Bank account of a customer for
// following operations: Deposit money | Withdraw Money | Show balance
contract demo{
mapping(address => uint) public user_account;
mapping(address => bool) public user_exist;

function create_account() public payable returns(string memory){
require(user_exist[msg.sender] == false, "Account Already created!");
user_account[msg.sender] = msg.value;
user_exist[msg.sender] = true;
return "Account created";
}

function deposit(uint amount) public payable returns(string memory){
require(user_exist[msg.sender] == true, "Account not created!");
require(amount > 0, "Amount should be greater than 0");
user_account[msg.sender] += amount;
return "Amount deposisted sucessfully";
}

function withdraw(uint amount) public payable returns(string memory){
require(user_exist[msg.sender] == true, "Account not created!");
require(amount > 0, "Amount should be greater than 0");
require(user_account[msg.sender] >= amount, "Amount is greater than money deposisted");
user_account[msg.sender] -= amount;
return "Amount withdrawn sucessfully";
}

function account_balance() public view returns(uint){
return user_account[msg.sender];
}
function account_exists() public view returns(bool){
return user_exist[msg.sender];
}

}
```

Output –


DEPLOY & RUN TRANSACTIONS
Solidity compiler
SmartContract.sol
Deploy
Publish to IPFS
At Address
Transactions recorded
Deployed Contracts
DEMO AT 0XD4F...2CBEE (MEMORY)
Balance: 0 ETH
create_account
deposit
withdraw
account_balan...
account_exists
user_account
user_exist
Low level interactions
CALLDATA
Transact
// SPDX-License-Identifier: GPL-3.0
pragma solidity >=0.8.2 <0.9.0;
// Write a smart contract on a test network, for Bank account of a customer for
// following operations: Deposit money | Withdraw Money | Show balance
contract demo{
mapping(address => uint) public user_account;
mapping(address => bool) public user_exist;
function create_account() public payable returns(string memory){
require(user_exist[msg.sender] == false, "Account Already created!");
user_account[msg.sender] = msg.value;
user_exist[msg.sender] = true;
return "Account created";
}
function deposit(uint amount) public payable returns(string memory){
require(user_exist[msg.sender] == true, "Account not created!");
require(amount > 0, "Amount should be greater than 0");
user_account[msg.sender] += amount;
return "Amount deposisted sucessfully";
}
function withdraw(uint amount) public payable returns(string memory){
require(user_exist[msg.sender] == true, "Account not created!");
require(amount > 0, "Amount should be greater than 0");
require(user_account[msg.sender] >= amount, "Amount is greater than money deposisted");
user_account[msg.sender] -= amount;
return "Amount withdrawn sucessfully";
}
function account_balance() public view returns(uint){
return user_account[msg.sender];
}
function account_exists() public view returns(bool){
return user_exist[msg.sender];
}
}


DEPLOY & RUN TRANSACTIONS
Publish to IPFS
At Address
Solidity compiler
Transactions recorded
Deployed Contracts
DEMO AT 0XD4F...2CBEE (MEMORY)
Balance: 0 ETH
create_account
deposit
amount: "10000"
Calldata
Parameters
transact
withdraw
amount: 4000
account_balan...
0: uint256: 6000
account_exists
0: bool: true
user_account
user_exist
Low level interactions
CALLDATA
Transact
// SPDX-License-Identifier: GPL-3.0
pragma solidity >=0.8.2 <0.9.0;
// Write a smart contract on a test network, for Bank account of a customer for
// following operations: Deposit money | Withdraw Money | Show balance
contract demo{
mapping(address => uint) public user_account;
mapping(address => bool) public user_exist;
function create_account() public payable returns(string memory){
require(user_exist[msg.sender] == false, "Account Already created!");
user_account[msg.sender] = msg.value;
user_exist[msg.sender] = true;
return "Account created";
}
function deposit(uint amount) public payable returns(string memory){
require(user_exist[msg.sender] == true, "Account not created!");
require(amount > 0, "Amount should be greater than 0");
user_account[msg.sender] += amount;
return "Amount deposisted sucessfully";
}
function withdraw(uint amount) public payable returns(string memory){
listen on all transactions
Search with transaction hash or address
[vm] from: 0x5B3...eddC4 to: demo.create_account() 0xD4F...2cbee value: 0 wei data: 0x509...f8633 logs: 0 hash: 0xb7b...ff82e
transact to demo.deposit pending ...
[vm] from: 0x5B3...eddC4 to: demo.deposit(uint256) 0xD4F...2cbee value: 0 wei data: 0xb6b...02710 logs: 0 hash: 0xb8c...cfa8f
transact to demo.withdraw pending ...
[vm] from: 0x5B3...eddC4 to: demo.withdraw(uint256) 0xD4F...2cbee value: 0 wei data: 0x2e1...00fa0 logs: 0 hash: 0x09d...19710
call to demo.account_balance
[call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: demo.account_balance() data: 0xc08...b9399
call to demo.account_exists
[call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: demo.account_exists() data: 0x33f...5915b
Debug