ASSIGNMENT 1

```
// SPDX-License-Identifier: MIT
pragma solidity >=0.7.0 <0.9.0;
contract bank{
    address public owner;
    uint public balance;

    modifier onlyOwner(){
        owner=msg.sender;
        _;
    }
    constructor(){
        owner=msg.sender;
    }
    function deposit(uint amount) public onlyOwner{
        require(amount>0);
        balance=balance+amount;
    }
    function withdraw(uint amount) public onlyOwner{
        require(amount>0);
        require(balance>=amount);
        balance=balance-amount;
    }
    function getbalance() public view returns (uint){
        return balance;
    }
    }
```

Explain above code in simple words

This code is a simple smart contract written in Solidity, which is a programming language for creating Ethereum-based contracts. Let's break it down in simple terms:

1. pragma solidity >=0.7.0 <0.9.0;`: This line specifies the version of Solidity that the contract is compatible with.

2. contract bank {`: This line defines a smart contract called "bank."

3. address public owner;`: It declares a variable called "owner" to store the Ethereum address of the owner of the contract. The "public" keyword means this variable can be read by anyone.

4. uint public balance;`: This declares a variable "balance" to store the balance of the bank. It's also public and can be read by anyone.

5. modifier onlyOwner() { ... }`: This is a custom modifier called "onlyOwner." Modifiers are used to add conditions to functions. In this case, it sets the "owner" variable to the address of the sender (the person interacting with the contract) and then allows the function with this modifier to be executed.

6. constructor() { ... }`: This is the constructor function, which runs only once when the contract is created. It sets the "owner" to the address that deploys the contract.

7. function deposit(uint amount) public onlyOwner { ... }`: This is a function called "deposit" that allows the owner to deposit a specified amount of money into the contract. It checks if the amount is greater than 0 and then adds it to the "balance."

8. `function withdraw(uint amount) public onlyOwner { ... }`: This is a function called "withdraw" that allows the owner to withdraw a specified amount of money from the contract. It checks if the amount is greater than 0 and if the contract's balance is sufficient to cover the withdrawal. If both conditions are met, it subtracts the amount from the balance.

9. `function getbalance() public view returns (uint) { ... }`: This is a function called "getbalance" that allows anyone to check the current balance of the contract without modifying it.

In summary, this smart contract simulates a simple bank account where the owner can deposit and withdraw funds, and others can check the balance. It uses a custom modifier to ensure that only the owner can perform deposit and withdrawal operations.

The provided code is a simple smart contract written in Solidity, a programming language for Ethereum blockchain. This contract is a basic representation of a bank. Here's a simplified explanation and some common questions about the code:

****Explanation****:

- This smart contract represents a bank and allows you to deposit and withdraw funds.

- The contract has an 'owner' (the person who deploys the contract) and keeps track of the 'balance' in the bank.

****Common Questions****:

1. ****What does this smart contract do?**

- This contract acts like a bank where you can deposit and withdraw funds. It keeps track of the account balance.

2. ****Who can interact with this contract?**

- Only the 'owner' of the contract can deposit or withdraw funds.

3. **How is the owner set?**

- The owner is initially set to the person who deploys the contract in the constructor.

4. **How can you deposit money into the bank?**

- You can use the 'deposit' function, providing an amount to deposit. The 'amount' must be greater than 0.

5. **How can you withdraw money from the bank?**

- You can use the 'withdraw' function, providing an amount to withdraw. The 'amount' must be greater than 0, and the balance must be sufficient to cover the withdrawal.

6. **How can you check the balance in the bank?**

- You can use the 'getbalance' function, which returns the current balance.

7. **What's the purpose of the `modifier onlyOwner`?**

- This modifier ensures that only the owner (the person who deployed the contract) can execute the functions it is applied to.

8. **What license is this code using?**

- The code is using the MIT license, which means it can be used and modified with certain conditions.

9. **What versions of Solidity is this code compatible with?**

- This code is compatible with Solidity versions greater than or equal to 0.7.0 and less than 0.9.0.