**Solidity Functions and Reentrancy Attack**

### **Task description:**

Implement a Vault contract and follow the provided specifications for the withdrawSafe() and withdrawUnsafe() functions. Create an Attacker contract and follow the provided specifications for the depositToVault() and attack() functions.

**Contract Vault**

Implement the following functions on the Vault contract as described in the task description:

* + withdrawSafe(address payable holder) - should send **via call** to the holder ether that they deposited to the contract (balance mapping).  
      
    **Must revert** in case of reentrancy attack.   
    Must reset the *holder’s* balance mapping after execution to zero.
  + withdrawUnsafe(address payable holder) - should send **via call** to the holder ether that they deposited to the contract.   
      
    **Must** **allow** a reentrancy attack.   
    Must reset the holder’s address balance after execution to zero.

### **Contract Attacker**

Implement the following functions as described:

* + attack() and fallback() - must withdraw all ether from Vault, using the withdrawUnsafe() function including ether that is not deposited by the attacker's contract.

## **What to hand in:**

1. Deploy the Vault contract.
2. Deploy the Attacker contract.
3. Deposit additional ether to Vault **not** from the Attacker contract. **No more than 0.001 ether.**
4. In the [Validator](https://sepolia.etherscan.io/address/0x70A003CaE209c00cB569fd41FAcE47608DA23d0f) contract, call validate(address vault, address attacker) to check the correctness of task implementation.
5. Provide the links both to the validated Vault and Attacker contracts.