BT PROGRAM 1

```
//SPDX-License-Identifier: UNLICENSED
pragma solidity ^0.8.0;

contract Bank{
  address public accOwner;
  uint256 balance=0;
  constructor(){
    accOwner=msg.sender;
  }

  function Deposit() public payable{
    require(accOwner==msg.sender,"You are not an account owner!!");
    require(msg.value > 0, "Amount should be greater than 0.");
    balance+=msg.value;
  }

  function Withdraw() public payable {
    require(accOwner==msg.sender,"You are not an account owner");
    require(msg.value > 0, "Amount should be greater than 0.");
    require(msg.value <= balance,"Account doesnot have sufficient balance!");
    balance-=msg.value;
  }

  function showBalance() public view returns(uint256){
    require(accOwner==msg.sender,"You are not an account owner");
    return balance;
  }
}
```

BT PROGRAM 2

```
// SPDX-License-Identifier: UNLICENSED
pragma solidity ^0.8.0;

contract StudentData {

  struct Student {
    string name;
    uint rollno;
  }

  Student[] public studentArr;

  // Add a new student
  function addStudent(string memory name, uint rollno) public {
    for (uint i = 0; i < studentArr.length; i++) {
      if (studentArr[i].rollno == rollno) {
        revert("Student with this roll number already exists!");
      }
    }
    studentArr.push(Student(name, rollno));
  }

  // Get number of students
  function getLengthOfStudents() public view returns (uint) {
    return studentArr.length;
  }

  // Display all students
  function displayAllStudents() public view returns (Student[] memory) {
    return studentArr;
  }

  // Get student by index
  function getStudentByIndex(uint index) public view returns (Student memory) {
    require(index < studentArr.length, "Index out of bound");
    return studentArr[index];
  }

  // Fallback function
  fallback() external payable {
    // This runs when someone calls a function that doesn't exist
    // You can use this to log or handle unexpected calls
    // Example: store received ether in contract
  }

  // Receive function
  receive() external payable {
    // This runs when contract receives plain Ether (no data)
    // Example: can log or store amount
  }
}
```