0xBlockPay Fuzzing Exercise

This is a short and simple exercise to start sharpening your smart contract fuzzing skills with Foundry.

The scenario is simple. There's a registry contract that allows callers to register by paying a fixed fee in ETH. If the caller sends too little ETH, execution should revert. If the caller sends too much ETH, the contract should give back the change.

Things look good according to the unit test we coded in the `Registry.t.sol` contract.

Your goal is to code at least one fuzz test for the `Registry` contract. By following the brief specification above, the test must be able to detect a bug in the `register` function.

Main contract

```
// SPDX-License-Identifier: UNLICENSED
pragma solidity ^0.8.13;

contract Registry {
    error PaymentNotEnough(uint256 expected, uint256 actual);

    uint256 public constant PRICE = 1 ether;

    mapping(address account => bool registered) private registry;

    function register() external payable {
        if(msg.value < PRICE) {
            revert PaymentNotEnough(PRICE, msg.value);
        }

        registry[msg.sender] = true;
    }

    function isRegistered(address account) external view returns (bool)
        {
        return registry[account];
    }
}
```

Test

```
// SPDX-License-Identifier: UNLICENSED
pragma solidity ^0.8.13;

import {Test, console2} from "forge-std/Test.sol";
import {Registry} from "../src/Registry.sol";

contract RegistryTest is Test {
    Registry registry;
    address alice;
```

```
10
11      function setUp() public {
12          alice = makeAddr("alice");
13
14          registry = new Registry();
15      }
16
17      function test_register() public {
18          uint256 amountToPay = registry.PRICE();
19
20          vm.deal(alice, amountToPay);
21          vm.startPrank(alice);
22
23          uint256 aliceBalanceBefore = address(alice).balance;
24
25          registry.register{value: amountToPay}();
26
27          uint256 aliceBalanceAfter = address(alice).balance;
28
29          assertTrue(registry.isRegistered(alice), "Did not register user
                ");
30          assertEq(address(registry).balance, registry.PRICE(), "
                Unexpected registry balance");
31          assertEq(aliceBalanceAfter, aliceBalanceBefore - registry.PRICE
                (), "Unexpected user balance");
32      }
33
34      /** Code your fuzz test here */
35  }
```

Solution

```
1   function test_fuzz_register(uint256 amountToPay) public {
2           vm.assume(amountToPay >= 1 ether);
3
4           vm.deal(alice, amountToPay);
5           vm.startPrank(alice);
6
7           uint256 aliceBalanceBefore = address(alice).balance;
8
9           registry.register{value: amountToPay}();
10
11          uint256 aliceBalanceAfter = address(alice).balance;
12
13          assertTrue(registry.isRegistered(alice), "Did not register user
                ");
14          assertEq(address(registry).balance, registry.PRICE(), "
                Unexpected registry balance");
15          assertEq(aliceBalanceAfter, aliceBalanceBefore - registry.PRICE
                (), "Unexpected user balance");
16      }
```

FUZZ Test

Step-by-Step Guide to Fuzz Testing in Forge

Set Up Your Project: Ensure you have Foundry installed. If not, you can install it by following the instructions on the Foundry Book.

1. Initialize a new Foundry project:

```
forge init my-fuzz-test
cd my-fuzz-test
```

2. Write Your Smart Contract: Create a simple smart contract to test. For example, a contract that allows deposits and withdrawals:

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

contract SimpleBank {
    mapping(address => uint256) public balances;

    function deposit() external payable {
        balances[msg.sender] += msg.value;
    }

    function withdraw(uint256 amount) external {
        require(balances[msg.sender] >= amount, "Insufficient balance")
            ;
        balances[msg.sender] -= amount;
        payable(msg.sender).transfer(amount);
    }
}
```

3. Write Fuzz Tests: Create a test file in the test directory, e.g., SimpleBank.t.sol

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

import "forge-std/Test.sol";
import "../src/SimpleBank.sol";

contract SimpleBankTest is Test {
    SimpleBank bank;

    function setUp() public {
        bank = new SimpleBank();
    }

```

```
    function testFuzz_DepositWithdraw(uint256 amount) public {
        // Assume the amount is within a reasonable range
        amount = bound(amount, 1, 1e18);

        // Deposit the amount
        bank.deposit{value: amount}();

        // Check the balance
        assertEq(bank.balances(address(this)), amount);

        // Withdraw the amount
        bank.withdraw(amount);

        // Check the balance again
        assertEq(bank.balances(address(this)), 0);
    }
}
```

4. Set foundry's toml file.

```
[fuzz]
# Specific settings for fuzz tests
runs = 1000  # Number of runs for each fuzz test
max_depth = 256  # Maximum call depth
```

5. Run the Tests: Execute the tests using the Forge command:

```
forge test
```