

## Bugs de Transferência

1º – Transferência com quantia negativa.

```
transfer(Alice, Charlie, -50)
----- Expected -----
Balances:
  Alice: 0
  Bob: 0
  Charlie: 0
Investments:
Next ID: 0
Error: Amount should be greater than zero
----- Actual -----
Balances:
  Alice: 50
  Bob: 0
  Charlie: -50
Investments:
Next ID: 0
Error:
-----

Assertion failed: are_equal(bank_state, expected_bank_state), file test.cpp, line 204
```

Alice faz uma transferência para Charlie no valor de -50, ocasionando num acréscimo de saldo para Alice e um débito no saldo do Charlie

Correção: No método 'transfer' foi adicionada uma validação para a quantia da transferência ser maior que 0.

```
29 29
30 30     string transfer(BankState &bank_state, string sender, string receiver,
31 31                          int amount) {
32 +   if (amount <= 0)
33 +   return "Amount should be greater than zero";
```

2º Saldo insuficiente para a transferência

Charlie faz uma transferência de 6 mas seu saldo é 0

```
transfer(Charlie, Charlie, 6)
----- Expected -----
Balances:
  Alice: 0
  Bob: 0
  Charlie: 0
Investments:
Next ID: 0
Error: Balance is too low
----- Actual -----
Balances:
  Alice: 0
  Bob: 0
  Charlie: 0
Investments:
Next ID: 0
Error:
-----
Assertion failed: error == expected_error, file test.cpp, line 205
```

Solução: Adicionado validação para que o saldo da conta seja maior ou igual ao valor da transferência.

```
26 + string transfer(BankState &bank_state, string sender, string receiver, int amount) {
27 +   if (bank_state.balances[sender] < amount)
28 +     return "Balance is too low";
29 + }
```

## Bugs de compra de investimento

1º – Compra de investimento sem saldo suficiente

```
buy_investment(Bob, 83)
----- Expected -----
Balances:
  Alice: 0
  Bob: 0
  Charlie: 0
Investments:
Next ID: 0
Error: Balance is too low
----- Actual -----
Balances:
  Alice: 0
  Bob: -83
  Charlie: 0
Investments:
  ID 0: { owner: Bob, amount: 83 }
Next ID: 1
Error:
-----

Assertion failed: are_equal(bank_state, expected_bank_state), file test.cpp, line 204
```

Bob com um saldo de 0 faz a compra de um investimento de 83. Deixando o saldo da sua conta negativo.

Solução:

```
43 + if (bank_state.balances[buyer] < amount)
44 +   return "Balance is too low";
45 +
```

## 2º - Compra de investimento com valor negativo

```
buy_investment(Bob, -42)
----- Expected -----
Balances:
  Alice: 0
  Bob: 0
  Charlie: 0
Investments:
Next ID: 0
Error: Amount should be greater than zero
----- Actual -----
Balances:
  Alice: 0
  Bob: 42
  Charlie: 0
Investments:
  ID 0: { owner: Bob, amount: -42 }
Next ID: 1
Error:
-----
Assertion failed: are_equal(bank_state, expected_bank_state), file test.cpp, line 204
```

Bob faz a compra de um investimento com saldo negativo

Solução:

```
33 33      string buy_investment(BankState &bank_state, string buyer, int amount) {
34 +      if (amount <= 0)
35 +          return "Amount should be greater than zero";
36 +      }
```

## Bugs de Depósito

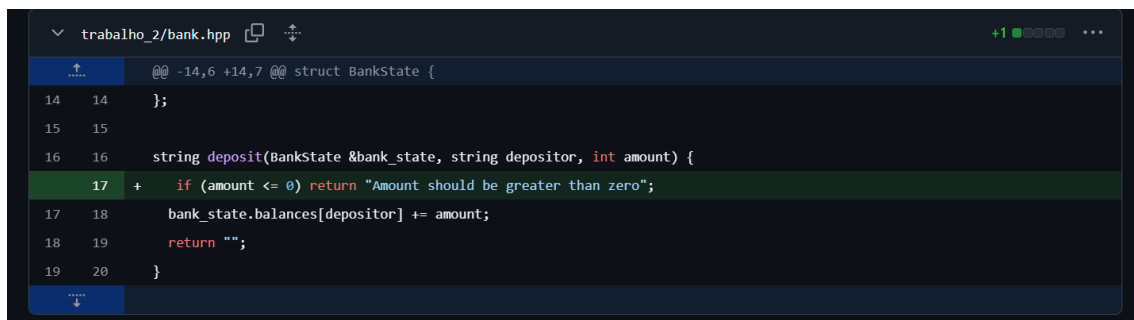
1º – Quantia de depósito menor ou igual a 0.

```
deposit(Alice, -30)
----- Expected -----
Balances:
  Alice: 0
  Bob: 0
  Charlie: 0
Investments:
Next ID: 0
Error: Amount should be greater than zero
----- Actual -----
Balances:
  Alice: -30
  Bob: 0
  Charlie: 0
Investments:
Next ID: 0
Error:
-----

Assertion failed: are_equal(bank_state, expected_bank_state), file test.cpp, line 204
```

Alice faz um depósito com valor negativo

Solução: Adicionado validação para que depósitos sejam feitos somente com valores maiores que 0.



```
trabalho_2/bank.hpp @@ -14,6 +14,7 @@ struct BankState {
14 14 };
15 15
16 16 string deposit(BankState &bank_state, string depositor, int amount) {
17 + if (amount <= 0) return "Amount should be greater than zero";
17 18 bank_state.balances[depositor] += amount;
18 19 return "";
19 20 }
```

## Bugs de Saque

1º - Saque com valor menor ou igual a 0

Alice tentou realizar um saque em sua conta com valor negativo.

```
withdraw(Alice, -7)
----- Expected -----
Balances:
  Alice: 0
  Bob: 0
  Charlie: 0
Investments:
Next ID: 0
Error: Amount should be greater than zero
----- Actual -----
Balances:
  Alice: 7
  Bob: 0
  Charlie: 0
Investments:
Next ID: 0
Error:

Assertion failed: are_equal(bank_state, expected_bank_state), file test.cpp, line 204
```

Solução: Adicionado uma validação para não permitir saques com valores negativos.

```
20 20 }
21 21
22 22 string withdraw(BankState &bank_state, string withdrawer, int amount) {
23 +   if (amount <= 0)
24 +     return "Amount should be greater than zero";
25 +
23 26   bank_state.balances[withdrawer] -= amount;
24 27   return "";
25 28 }
```

2º - Saque com saldo insuficiente

Charlie tentou sacar uma quantia de 30, mas seu saldo na conta era de 0.

```

withdraw(Charlie, 30)
----- Expected -----
Balances:
  Alice: 0
  Bob: 0
  Charlie: 0
Investments:
Next ID: 0
Error: Balance is too low
----- Actual -----
Balances:
  Alice: 0
  Bob: 0
  Charlie: -30
Investments:
Next ID: 0
Error:
-----
Assertion failed: are_equal(bank_state, expected_bank_state), file test.cpp, line 204

```

Solução: Adicionado validação para não permitir saques com valores superiores ao saldo da conta.

↑	....	@@ -23,6 +23,9 @@ string withdraw(BankState &bank_state, str
23	23	<code>if (amount &lt;= 0)</code>
24	24	<code>return "Amount should be greater than zero";</code>
25	25	
26	+	<code>if(bank_state.balances[withdrawer] &lt; amount)</code>
27	+	<code>return "Balance is too low";</code>
28	+	
26	29	<code>bank_state.balances[withdrawer] -= amount;</code>
27	30	<code>return "";</code>
28	31	<code>}</code>

## Bugs de Venda Investimento

1º Charlie vendeu um investimento que não existe

```
sell_investment(Charlie, 1)
----- Expected -----
Balances:
  Alice: 0
  Bob: 0
  Charlie: 0
Investments:
Next ID: 0
Error: No investment with this id
----- Actual -----
Balances:
  Alice: 0
  Bob: 0
  Charlie: 0
Investments:
  ID 1: { owner: , amount: 0 }
Next ID: 0
Error:
-----
Assertion failed: are_equal(bank_state, expected_bank_state), file test.cpp, line 204
```

Solução: Adicionado uma validação para não permitir vender investimentos que não existem.

```
55 + string sell_investment(BankState &bank_state, string seller, int investment_id) {
56 +   if (bank_state.investments.find(investment_id) == bank_state.investments.end())
57 +     return "No investment with this id";
58 +
59   bank_state.balances[seller] -= bank_state.investments[investment_id].amount;
60   return "";
61 }
```

2º - Venda de investimento de não autoria

Charlie vendeu um investimento que não lhe pertencia



```

sell_investment(Charlie, 0)
----- Expected -----
Balances:
  Alice: 0
  Bob: 51
  Charlie: 0
Investments:
  ID 0: { owner: Bob, amount: 3 }
Next ID: 1
Error: Seller can't sell an investment they don't own
----- Actual -----
Balances:
  Alice: 0
  Bob: 51
  Charlie: 3
Investments:
  ID 0: { owner: Bob, amount: 3 }
Next ID: 1
Error:
-----

Assertion failed: are_equal(bank_state, expected_bank_state), file test.cpp, line 204

```

Solução: Adicionado uma validação para não permitir vender investimentos que não pertence ao próprio vendedor

3º - Ao vender um investimento, o investimento não era removido da lista de investimentos do cliente.

Bob, ao vender o investimento de id 0, não teve o investimento removido de sua conta

```

sell_investment(Bob, 0)
----- Expected -----
Balances:
  Alice: 0
  Bob: 99
  Charlie: 0
Investments:
  Next ID: 1
Error:
----- Actual -----
Balances:
  Alice: 0
  Bob: 99
  Charlie: 0
Investments:
  ID 0: { owner: Bob, amount: 36 }
Next ID: 1
Error:
-----

Assertion failed: are_equal(bank_state, expected_bank_state), file test.cpp, line 204

```

Solução: Adicionado lógica para remover o investimento do cliente ao realizar uma venda.

```
63 63      return "Seller can't sell an investment they don't own";
64 64
65 +   bank_state.investments.erase(investment_id);
66 +
65 67     bank_state.balances[seller] += bank_state.investments[investment_id].amount;
66 68     return "";
67 69 }
```