BLOCK NONCE #4: Don't make your tests too DRY

by guy.f@

What?

The DRY principle ("Don't Repeat Yourself") encourages code reuse rather than duplication. But is it also the best thing for tests?

Is it clear what this test does just from reading its code?

```
#[test]
fn verify_balance_on_transfer() {
  let mut account = create_default_account();  // Test helper method.
  make_transfers(&mut account);                // Test helper method.
  assert_eq!(account.balance(), 1500);
}
```

Why?

While the test body above is concise, the reader cannot understand what it does without looking at other places. Since tests don't have tests, it should be easy for humans to manually inspect them for correctness, even at the expense of some code duplication.

How?

Make your tests **DAMP**! (Descriptive and Meaningful Phrases).
In the context of unit tests, this test would be easier to read and verify for correctness:

```
#[test]
fn verify_balance_on_transfer() {
  const INITIAL_BALANCE : u32 = 1000;
  const TRANSFER_AMOUNT : u32 = 500;

  let mut new_account =
      Account::new(AccountType::NO_OUTGOING_TRANSFERS(INITIAL_BALANCE));
  let mut account_to_transfer_from =
      Account::new(AccountType::SUPPORTS_TRANSFERS(TRANSFER_AMOUNT));

  assert_eq!(new_account.balance(), INITIAL_BALANCE);
  account_to_transfer_from.transfer_to(&mut new_account, TRANSFER_AMOUNT);
  assert_eq!(new_account.balance(), INITIAL_BALANCE + TRANSFER_AMOUNT);
}
```

TIP💡 The DRY principle is still relevant in tests. There is a trade-off between readable and unique. Try to avoid duplicate code (DRY) while keeping the code expressive and easy to verify (DAMP). If unsure, lean more heavily on the side of DAMP.