Report 2

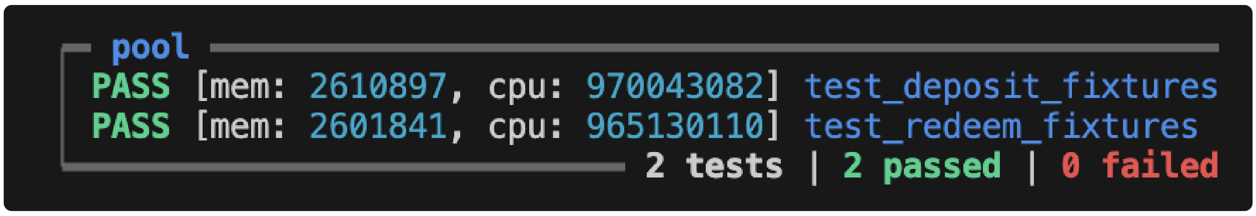
Test Results and Documentation

There are 2 complex tests for buy (deposit ADA into the curve -> receive token) and sell (return token into the curve and receive ADA). Since there is no random generators in Aiken the test flow was as follows:

- We reproduced all calculations and validations in Rust;
- We implemented generators of random pool state and buy/sell operations;
- We chose a random state and operations and transfer it to Aiken tests.

We use already implemented and audited limit order contracts for our orders, so the tests only checks the pool contract consistency.
For buy and sell operations test names are `test_deposit_fixtures` and `test_redeem_fixtures` respectively.
For a random pool state both tests verifies that:

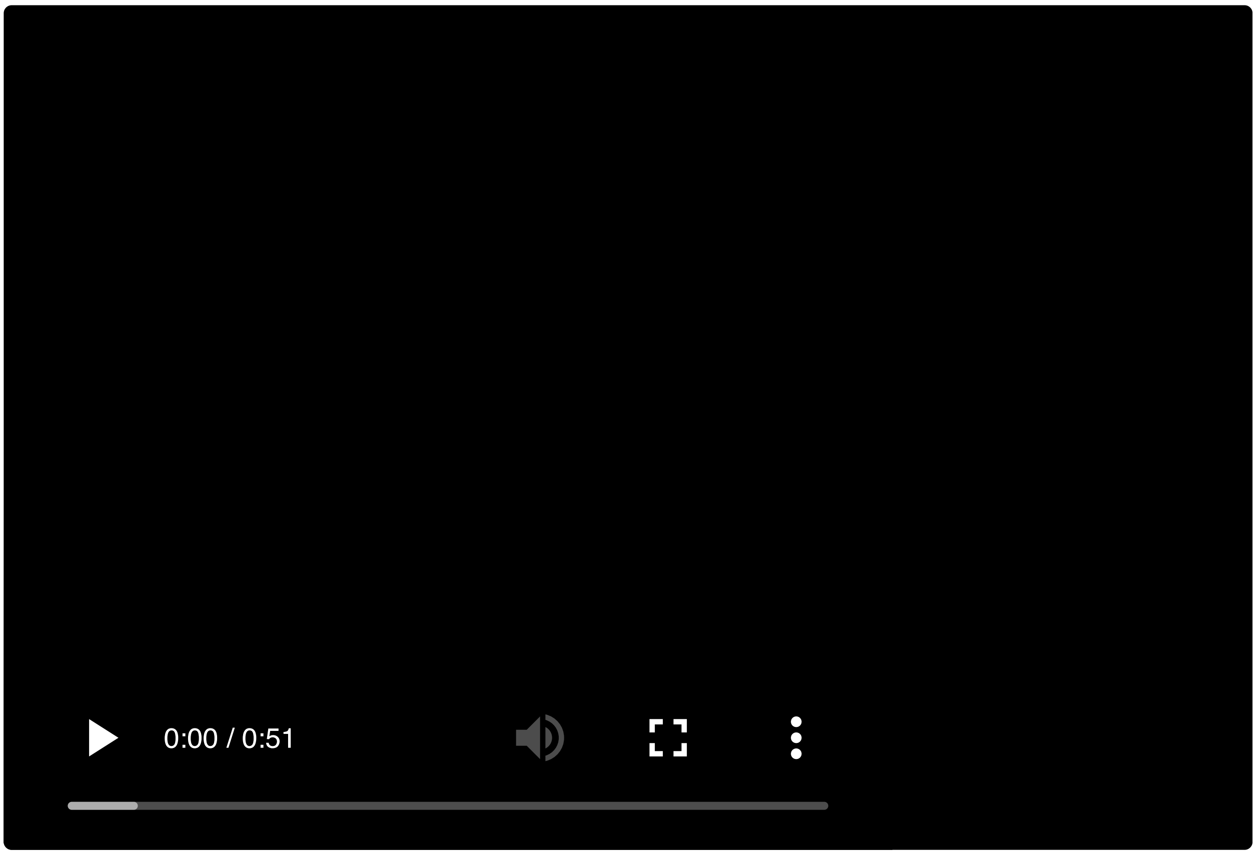
- Pool validator return true for the given input/output;
- Pool validators return false if the output is at least 1 token less than the expected output;
- Pool validators return false if the output is at least 1 lovelace less than the expected output;
- Pool validators return false if pool is drained;
- Pool validators return false if pool datum is changed.



```
┍━ pool ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━
│ PASS [mem: 2610897, cpu: 970043082] test_deposit_fixtures
│ PASS [mem: 2601841, cpu: 965130110] test_redeem_fixtures
┕━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2 tests | 2 passed | 0 failed
```

Operational Demonstration

The video attached



Explorer Links

Order creation tx:
https://cardanoscan.io/transaction/9e823bbf07e20e52a1933e80ee5a8487b6fa957e807ec6758fc8592c293fd06a?tab=utxo
Order execution: tx:
https://cardanoscan.io/transaction/1b8cfdda563c7959300fbc1163d9ad7033dfc1bfc6815f0b7bfa5f4d44c05f5b?tab=utxo