BLOCK NONCE #7: Snapshot Testing - Without Making Everyone Angry

What?

Test that values equal some hard-coded number that rarely changes, without causing other devs to come at you with torches and pitchforks if they need to fix the values.

Why?

Some values should not change unless explicitly expected to; for example, gas cost of a fixed transaction, unless compiler version is updated.

How?

Use the `expect-test` crate, to hard-code values that are auto-fixable! On failure, the command to fix the snapshot values is printed to console.

Instead of:

```
assert_eq!(computed_value, 7);
```

Do:

```
expect!["7"].assert_debug_eq(computed_value);
```

Or, if the expected output is in a file (and not inlined in code), do:

```
expect_file!["path_to_file.txt"].assert_debug_eq(computed_value);
```

TIP💡 Remember to take different test cases into account! Maybe one `expect!` is not enough...

TIP💡 For comparing large objects, consider the specific snapshotted field separately. For example, instead of:

```
let expected_tx_result = <HARD CODED TX RESULT>;
let tx_result = run_tx();
assert_eq!(tx_result, expected_tx_result);
```

Do:

```
let mut expected_tx_result = <HARD CODED TX RESULT>;
let mut tx_result = run_tx();
expect!["7"].assert_debug_eq(tx_result.gas_cost);
// Overwrite the single snapshot field.
expected_tx_result.gas_cost = 0;
tx_result.gas_cost = 0;
assert_eq!(tx_result, expected_tx_result);
```