Homework 11 Answers

1. Install avm and verify the CLI is installed properly `anchor --version`

```
anchor init storage # creates new anchor project called storage
```

2. Program

```
use anchor_lang::prelude::*;

declare_id!("4dWrCZRqvr2xAFkw6CqKr5qCadzps5Aooq4aBgYczvwM");

#[program]
pub mod storage {
use super::*;

        pub fn initialize(ctx: Context<Initialize>, balance: u64) -> Result<()> {

ctx.accounts.storage_account.balance = balance;
                msg!("Changed balance to:
```

```
{}!", balance);
                Ok(())
        }
}

#[derive(Accounts)]
pub struct Initialize<'info> {
        #[account(
                init,
                seeds=
[signer.key().as_ref()],
                bump,
                payer = signer,
                space = 8 + 8
        )]
        pub storage_account: Account<'info,
StorageData>,
        #[account(mut)]
        pub signer: Signer<'info>,
        pub system_program: Program<'info,
System>,
}

#[account]
pub struct StorageData {
        balance: u64,
}
```

3. Tests

```
import * as anchor from '@coral-xyz/anchor';
import { Program } from '@coral-xyz/anchor';
import { Storage } from
'../target/types/storage';
import { expect } from 'chai';
import { before } from 'mocha';

// Configure the client to use the local
cluster.
const provider =
anchor.AnchorProvider.env();
anchor.setProvider(provider);

const program = anchor.workspace.Storage as
Program<Storage>;
const systemProgram =
anchor.web3.SystemProgram.programId;
let storageDataPDA;
describe('storage', async () => {
        it('Intialize', async () => {
                const data = new
anchor.BN(100);
                [storageDataPDA] = await
anchor.web3.PublicKey.findProgramAddressSync
(
[provider.publicKey.toBuffer()],
                        program.programId
                );
                const tx = await
program.methods
                        .initialize(data)
                        .accounts({

storageAccount: storageDataPDA,

systemProgram: systemProgram,
                        })
                        .signers([])
                        .rpc();
                console.log('🚀
Intialization transaction:', tx);
        });

        it('Account data is initialized to
100', async () => {
                // fetch data for pda
                const data = await
program.account.storageData.fetch(storageDat
aPDA);
                // convert BN to decimal
                const value =
parseInt(data.balance.toString('hex'), 16);
                expect(value).equal(100,
'Value is not 100');
         });
});
```

See repo.