**Practical – 5**

**Aim: Write a program that will demonstrates adding blocks to the blockchain**

**Code:**

***tests.rs***

#[allow(dead\_code)]

pub fn blockchain\_class\_print\_blockchain() {

let utsav = Client::new();

let bhupendra = Client::new();

let jash = Client::new();

let mut block0 = Block::genesis\_block(&utsav);

println!("utsav public key: {}", utsav.identify());

println!("bhupendra public key: {}", bhupendra.identify());

println!("jash public key: {}", jash.identify());

println!("");

let mut transaction1 = Transaction::new(utsav.public\_key, bhupendra.public\_key, 10.0, None);

transaction1.sign\_transaction(&utsav);

println!(

"Transaction 1 signature validation: {:#?}",

transaction1.is\_valid\_transaction()

);

block0.verified\_transactions.push(transaction1);

let mut transaction2 = Transaction::new(bhupendra.public\_key, jash.public\_key, 10.0, None);

transaction2.sign\_transaction(&bhupendra);

println!(

"Transaction 2 signature validation: {:#?}",

transaction2.is\_valid\_transaction()

);

block0.verified\_transactions.push(transaction2);

let mut transaction3 = Transaction::new(jash.public\_key, utsav.public\_key, 10.0, None);

transaction3.sign\_transaction(&jash);

println!(

"Transaction 3 signature validation: {:#?}",

transaction3.is\_valid\_transaction()

);

block0.verified\_transactions.push(transaction3);

let \_last\_block\_hash = block0.calculate\_hash();

let mut coin\_chain = Blockchain::new();

coin\_chain.add\_block(block0);

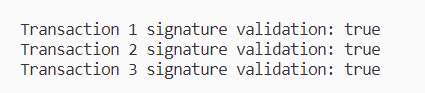
coin\_chain.dump\_blockchain();

}

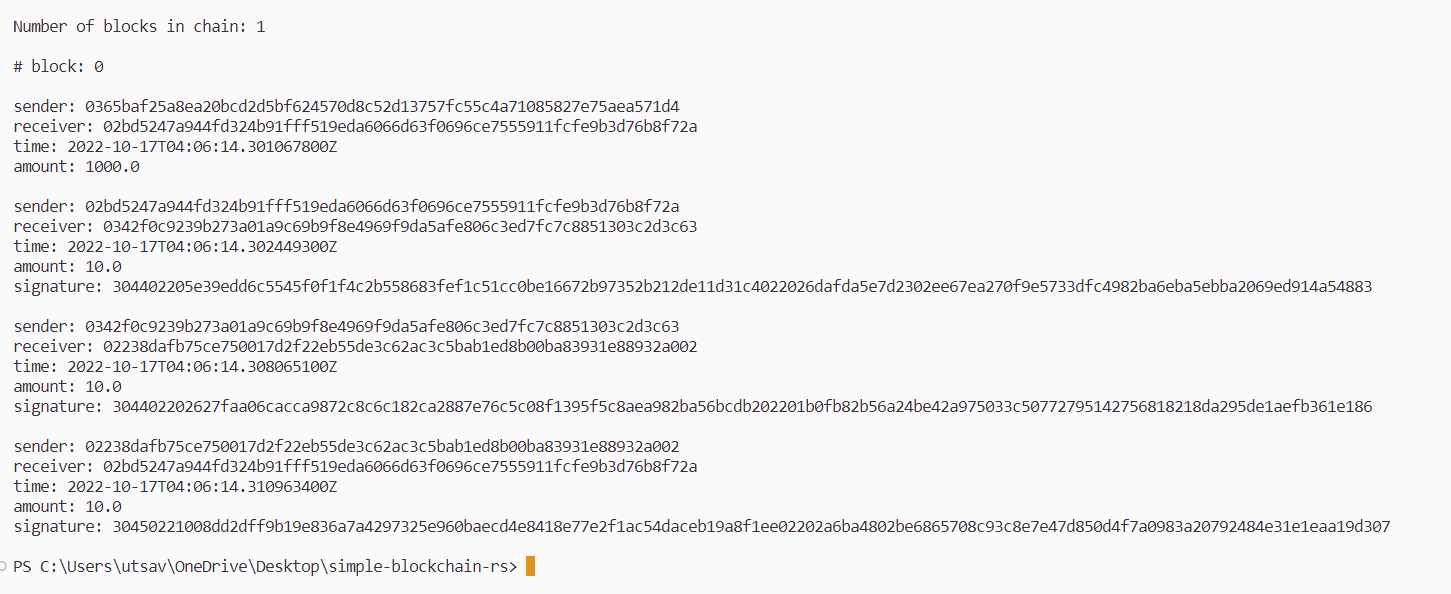
**Public keys**

****

**Transactions validation**

****

**Print blockchain**

****