**Practical – 7**

**Aim: Write a program that will add blocks in the blockchain application.**

**Code:**

/// This method prints the blockchain.

pub fn dump\_blockchain(&self) {

println!("\nNumber of blocks in chain: {}", &self.blocks.len());

for block in &self.blocks {

println!("\n# block: {}\n", block.index);

for t in &block.verified\_transactions {

t.print\_transaction();

}

}

}

/// Mining the block in blockchain

pub fn mine\_block(&mut self, difficulty\_level: usize) {

let mut nonce = 0;

loop {

let hash = crypto\_hash::hex\_digest(

crypto\_hash::Algorithm::SHA256,

format!("{}{}", self.serialize\_block(), nonce).as\_bytes(),

);

if hash.starts\_with(&DIFFICULTY\_STRING.repeat(difficulty\_level)) {

self.nonce = nonce;

self.block\_hash = hash;

println!("Block Mined!: {:#?}", self);

break;

}

if nonce > 10000 {

panic!("Difficulty is too high! block mining failed.");

}

nonce = nonce + 1;

}

}

}

/// Pushing block into the blockchain

pub fn push\_block\_into\_blockchain() {

let utsav = Client::new();

let bhupendra = Client::new();

let jash = Client::new();

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

block0.mine\_block(2);

let mut block1 = Block::new(1, &block0.block\_hash);

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);

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

transaction2.sign\_transaction(&bhupendra);

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

transaction3.sign\_transaction(&jash);

block1.verified\_transactions.push(transaction1);

block1.verified\_transactions.push(transaction2);

block1.verified\_transactions.push(transaction3);

block1.mine\_block(2);

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

balar\_chain.add\_block(block0);

balar\_chain.add\_block(block1);

balar\_chain.dump\_blockchain();

}

