Assignment 2:

**1)**

class turn{

private prevNo = Symbol();

private currNo = Symbol();

constructor(){

this.prevNo = 0;

this.currNo = 1;

}

next(){

let curr = Symbol();

curr = this.prevNo + this.currNo;

this.prevNo = this.currNo;

this.currNo = curr;

return {

current : curr,

next : this.prevNo + this.currNo

}

}

}

let t = new turn();

console.log(t.next());

console.log(t.next());

console.log(t.next());

console.log(t.next());

console.log(t.next());

2.

// we can push arms nos into a list and then use this

// here i have taken predefined list with numbers

const listArm=[0,1,2,3,4,5,6,7,8,9,153,370,371];

const iterator = listArm[Symbol.iterator]();

function getNextArmstrong(){

return {

"next value" : iterator.next().value

}

}

console.log(getNextArmstrong());

console.log(getNextArmstrong());

console.log(getNextArmstrong());

console.log(getNextArmstrong());

console.log(getNextArmstrong());

console.log(getNextArmstrong());

3.

// we can push arms nos into a list and then use this

// here i have taken predfined list with numbers

const listArm = [3,4,5,6,7,8,9,153,370,371,407,1634,8208,9474];

const iterator = listArm[Symbol.iterator]();

function getnext(){

return iterator.next().value;

}

// generator

function\* arms() {

let current = 1;

let next = 2;

while (true) {

let reset = yield current;

[current, next] = [next, getnext(next)];

if (reset) {

current = 1;

next = 2;

// arms();

}

}

}

const s = arms();

console.log(s.next().value); // 1

console.log(s.next().value); // 2

console.log(s.next().value); // ...

console.log(s.next().value);

console.log(s.next().value);

console.log(s.next().value);

console.log(s.next().value);

console.log(s.next().value);

console.log(s.next().value); // 9

// should return next ARMSTRONGs

console.log(s.next().value);

console.log(s.next().value);

console.log(s.next().value);

console.log(s.next().value);

console.log(s.next().value);

console.log(s.next().value);

console.log(s.next().value);

// reset generator

console.log(s.next(true).value);

console.log(s.next().value); // 1

console.log(s.next().value); // 2

Assignment 3:

1)

Promise.all([x, y, z])

.then(results => {

const total = results.reduce((p, c) => p + c);

console.log(`Results: ${results}`);

console.log(`Total: ${total}`);

});

2)

class Account{

constructor (public id: number , public name:string , public balance:number){

}

totalBalance(){

return this.balance;

}

class SavingsAccount extends Account{

constructor(id: number , name:string , balance:number, interest: number){

super(id ,name ,balance= balance+(balance\*interest)/100);

}

}

class CurrentAccount extends Account{

constructor(id: number , name:string , balance:number , cash\_credit: number){

super(id ,name ,balance= balance+cash\_credit);

}

}

let savingsAccount =new SavingsAccount(9,"ram",250000,10);

let currentAccount =new CurrentAccount(27,"ram",250000,1000);

console.log(savingsAccount.totalBalance());

console.log(currentAccount.totalBalance());

3)

interface Printable{

print():any;

}

let circle: Printable ={

print(){

console.log("circle obj");

}

}

let employee: Printable ={

print(){

console.log("employee obj");

}

}

function printAll(...objArr: Array<any>){

for(let i in objArr){

console.log(objArr[i].print())

}

}

4)