LAB # 7

Modern JavaScript (ES5\ES6\ESNEXT)

OBJECTIVE:

To get familiar with modern JavaScript and its features.

Lab Task:

Modern JS

a. Create a class called Library and add a few properties like sections, books,manager, and openingTime and closingTime.

b. Add few functions like manageLibrary, issueBooks, addNewSection, openLibrary, and closeLibrary.

c. Use arrow functions instead of regular functions for all function definition.

d. Return promise from issueBooks, openLibrary and closeLibrary functions.

e. Return books from issueBooks promise. Use then to receive the result.

f. Return openingTime from openLibrary function. Use setTimeout and resolve after 2 seconds. Use async/await keywords pair to get the results.

g. Return closingTime from closeLibrary function. Use setTimeout and resolve after 1 second. Use async/await keywords pair to get the results.

h. Create instances of the Library class and call each function as mentioned above.

i. Loop through the keys of each instance using for-in loop and log the keys.

Source Code:

```
class Library{
    constructor (section,book,manager,opentime,closetime){
        this.section=section;
        this.book=book;
        this.manager=manager;
        this.opentime=opentime;
        this.closetime=closetime;
    }
```

```
manageLibrary=()=>{
        console.log(`library manager ${this.manager}`)
    }
    issuebooks=()=>{
        return new Promise(function(myResolve,myReject){
            setTimeout(function(){
                myResolve(`issue books Java Script`);
            }, 2000)
        });
        myPromise.then((response)=>{
            console.log(response)
        })
    }
    addnewsection=()=>{
        console.log(`New Section of books ${this.section}`)
    }
    openLibrary=()=>{
        return new Promise(function(myResolve,myReject){
            setTimeout(function(){
                myResolve(`Open Time 12:30`);
            }, 2000)
        });
        myPromise.then((response)=>{
            console.log(response)
        })
    }
    closeLibrary=()=>{
        return new Promise(function(myResolve,myReject){
            setTimeout(function(){
                myResolve(`Close Time 1:00`);
            }, 2000)
        });
        myPromise.then((response)=>{
            console.log(response)
        })
    }
}

async function test(){
    var lib= new Library('A','Java Script', 'M. Talha Rehman', '12:30',
'1:00')
    lib.manageLibrary()
    lib.addnewsection()
    const j= await lib.openLibrary()
    const k= await lib.closeLibrary()
```

```
    const l= await lib.issuebooks()
    console.log(l)
    console.log(j)
    console.log(k)
}
test()
```

Output:

Home Task

Modern JS

a. Create a generator function called generateRegistrationNumbers.

i. Accept the parameter asking the upper limit of the Registration number to be generated.

ii. Calling the next method should return first registration number starting from 1

iii. Calling the next method of generator again should return the next number.

iv. If the limit has been reached, the generator should stop giving next number no matter how many more times you call the next method of generator.

Source Code:

```
function* generateRegistrationNumbers(value) {
    let i = 1;
    while (true) {
      yield i++;
    }
  }

  const ids = generateRegistrationNumbers();

  console.log(ids.next().value);
  console.log(ids.next().value);
  console.log(ids.next().value);
  console.log(ids.next().value);
  console.log(ids.next().value);
  console.log(ids.next().value);
```

Output: