German International University
Faculty of Informatics and Computer Science
Dr. Iman Awaad
Eng. Ahmed Sherif
Amir Haythem
Mohamed Essam
Yassein Eldamasy

Software Engineering, Winter 2025
Practice Assignment 5

Solution

Exercise 5-1

```
//main.js
function getReminder(x,y){
return x % y;
}

function isEven(x){
return getReminder(x,2) === 0;
}

function getEven(arr){
return arr.filter( num  => num % 2 === 0  );
}

function getOdd(arr){
return arr.filter( num  => num % 2 === 1  );
}
```

Assume that main.js and app.js are in the same directory

How can we import function isEven from main.js to app.js ?
How can we import function getEven and getOdd from main.js to app.js ?

Solution:

Export: in main.js file

```
function getRemainder(x,y){
    return x%y;
}
function isEven(x){
    return getRemainder(x,2)===0;
}
function getEven(arr){
    return arr.filter(num=> num%2===0)
}
function getOdd(arr){
    return arr.filter(num=> num%2===1)
}
// we will use module.exports and write name of functions inside {}
module.exports = {getEven,getOdd,isEven}
```

Import: in app.js file

```
const { isEven,getEven,getOdd } =require ("./main");

console.log(getEven([1,2,3,4]))
console.log(isEven(2))
```

Exercise 5-2

What is the output for the given code ?

```
const firstWeek = () => { console.log( "Week 1" ); }
const secondWeek = () => { console.log( "Week 2" ); }

setTimeout( firstWeek, 5000  ); // async Function
setTimeout( secondWeek, 4000  );

console.log("Week 3");
console.log("Week 4");
```

Solution:

Week 3
Week 4
Week 2
Week 1

explanation:
web API will be responsible for setTimeout so, week 3 and week 4 will be printed then 4 seconds later week 2 will be printed then week 1 will be printed 1 second later after week 2

Exercise 5-3

What is the output for the given code ?

```
const getReminder = ( x,y ) => { return x % y;   }

const getTypeNumber =  (x, callback) => {

const reminder = callback(x,2);
if( reminder == 0){
console.log(`${x} is an Even Number`);
}else{
console.log(`${x} is an odd Number`);
}

}

getTypeNumber(7,getReminder);
```

Solution:

7 is an odd Number

Exercise 5-4

What is the output for the given code ?

```
async function callTimer(){

setTimeout( () => {
  console.log("First Timer: 7:45");
  setTimeout (  () => {
 console.log("Second Timer 7:55");
 setTimeout( () => {
console.log("End");
setTimeout(  () => {
  console.log("Third Timer 8:05");
  },3000 )
},2000  )
},3000 )
},2000)

}

callTimer();
console.log("Start Timer");
```

Solution:

Start Timer
First Timer: 7:45
Second Timer: 7:55
End
Third Timer : 8:05

Note that
At time 2 seconds : First Timer: 7:45
At time 5 seconds : Second Timer 7:55
At time 7 seconds : End
At time 10 seconds : Third Timer 8:05

Exercise 5-5

What is the output for the given code ?

```
const isAppointment = false;

const appointment = (appointmentFlag) => { return new Promise( (resolve , reject) =>
{

if( appointmentFlag ){
reject( new Error("cannot schedule new appointment") );
} else {
const newAppointment = { name : "office hours" , timing : "8pm"};
resolve( newAppointment );
}
```

```
})}

const addDefaultDetails = (res) => {
res.location="Zoom";
return Promise.resolve(res);
}



appointment(isAppointment).then( res => { return addDefaultDetails(res); })
.then( res =>
console.log(‘Appointment ${res.name} on ${res.location} at ${res.timing}‘) )
.catch(err => console.log(err.message))
```

Assume that the value of isAppointment changed to true
What will be the ouptut for the above code ?

Solution:

```
In case that isAppointment is set to false
Appointment office hours on Zoom at 8pm will be printed

In case that isAppointment is set to true
cannot schedule new appointment will be printed
```

Exercise 5-6

New Function myAppointment was added to the code in Ex 5.5
What is the output for the given code ?

```
async function myAppointment(appointmentFlag){
try{
const appointmentDetails =  await appointment(appointmentFlag);
const res =  await addDefaultDetails(appointmentDetails);
console.log( ‘Appointment ${res.name} on ${res.location} at ${res.timing}‘ );
} catch(e) {
console.log("error message: ",e.message)
}
}

const flag = true;
myAppointment( flag );
```

Assume that the value of flag changed to false
What will be the ouptut for the above code ?

Solution:

```
In case that flag is set to false
Appointment office hours on Zoom at 8pm will be printed

In case that flag is set to true
cannot schedule new appointment will be printed
```

Exercise 5-7

```
Given the following asynchronous function called getStudents
which takes one argument :
1. studentId : string

getStudents return a promise object with the following properties:
1. studentId : string
2. GPA: Number
3. email : string
4. name : string
```

```
function getStudents(studentId){
    let studentObj = {name : "notFound"};
    switch(studentId){
        case "1002345": studentObj = {GPA : 0.7,
        email :"mo.ihab@student.giu-uni.de", name : "Mohamed Ihab"};break;
        case "1002340": studentObj = {GPA : 0.75,
        email :"nada.yasser@student.giu-uni.de", name : "Nada Yasser"};break;
        case "1002346": studentObj = {GPA : 0.79,
        email :"mostafa.saeed@student.giu-uni.de", name : "Mostafa Saeed"};break;
    }
    return studentObj.name!="notFound"?Promise.resolve(studentObj)
    :Promise.reject(new Error(‘student id ${studentId} doesn’t exists‘));

}
```

Write a function called displayStudentInfo which takes one argument :
1. students : array of string

returns array of string where each item of the output array should adhere to the following format :
name:email:GPA:id
Implement function displayStudentInfo twice
The first implementation using async/await approach
you are not allowed to use async/await in the second implementation
Note that students is array of students id

Solution:

```
 function getStudents(studentId){
    let studentObj = {name : "notFound"};
    switch(studentId){
        case "1002345": studentObj = {GPA : 0.7,
        email :"mo.ihab@student.giu-uni.de", name : "Mohamed Ihab"};break;
        case "1002340": studentObj = {GPA : 0.75,
        email :"nada.yasser@student.giu-uni.de", name : "Nada Yasser"};break;
        case "1002346": studentObj = {GPA : 0.79,
        email :"mostafa.saeed@student.giu-uni.de", name : "Mostafa Saeed"};break;
    }
    return studentObj.name!="notFound"?Promise.resolve(studentObj)
    :Promise.reject(new Error(‘student id ${studentId} doesn’t exists‘));

}

//First implementation using async await:


async function displayStudentInfoV1(students) {
  let result = [];
  for (let studentId of students) {
    try {
      let student = await getStudents(studentId);
      let info = ‘${student.name}:${student.email}:${student.GPA}:${studentId}‘;
      result.push(info);
    } catch (error) {
      result.push(error.message);
    }
  }
  return result
}

//Second implementation using promise:

function displayStudentInfoV2(studentArray) {
  let info = Promise.all(
    studentArray.map((StudentId) => {
      return getStudents(StudentId)
.then(student => ‘${student.name}:${student.email}:${student.GPA}:${StudentId}‘)
.catch(err => err.message);
    })
  );
  return info;
}

async function test(){
result = await displayStudentInfoV1(["1002345", "1002340", "1002346" , "234"]);
console.log("async await approach is" , result);
result = await displayStudentInfoV2(["1002345", "1002340", "1002346" , "234"]);
console.log("promise approach is" , result);
}

test();
```