# Instruction for second graded exercise on OO JS

You will have **90** minutes to write code that will pass all 10 provided tests.
File in which all test are written is called **functions.test.js**.
The code for test is hashed, so You will not understand it's details.
You have to create a file called **functions.js** and in this file create and export all functions that are need by the tests.
Both of those files have to be inside same folder, Jest and NPM force this.

This is the link to the Jest documentation. You can use it while working on code.

PLAGIARISM IS STRICTLY FORBIDDEN WILL BE TREATED AS FAILURE WITH NO POSSIBILITY TO ATTEND AT ANY RESIT FOR ALL INVOLVED .

## First test

Create class `Book` which will have properties:

- ISBN
- title
- author
- genre

In addition this class have an constructor in which all those properties are set.

## Second test

Create class `Shelf`.

`Shelf` should have property called `books` which is the array of stored books on shelf.

`Shelf` must have an function:

- addBook, pass 1 argument which will be a book that need to be added to this shelf

## Third test

Add to `Shelf` function:

- `removeBook` that get's an function as parameter which will remove all books from shelf that meets the given criteria

## Fourth test

Add to `Shelf` function:

- `getAllBy` that get's an function as parameter which will fetch all books by given criteria. Books that have matched those criteria are returned as new array. This function does not change any of books on given shelf.

## Fifth test

Create `SortedShelf` class that inherits all functions and attributes from `Shelf`. In constructor of `SortedShelf` sorting function must be provided and saved as attribute of this class in name of `sortingFunction`.
When function `addBook` is called whole collection of books have to be sorted by `sortingFunction`.

## Sixth test

Create `Library` class that have:

- `shelfs` -> array of shelfs in library
- function `addShelf` -> which get 1 parameter and will add a given shelf to `shelfs`
- function `getBookByAuthor` -> which will return all books from all `shelfs` that meets the criteria of passed string argument. This function does not remove any books from `shelfs`
- function `getBookByISBN`-> which will return all books from all `shelfs` that meets the criteria of passed int argument.
- function `getBookByGenre`-> which will return all books from all `shelfs` that meets the criteria of passed string argument.
- function `getBookByTitle`-> which will return all books from all `shelfs` that meets the criteria of passed string argument.

## Seventh test

To `Library` class add:

- function `getBookByAuthorRegex` -> which will return all books from all `shelfs` that meets the criteria of passed regex argument,
- function `getBookByGenreRegex` -> which will return all books from all `shelfs` that meets the criteria of passed regex,
- function `getBookByTitleRegex` -> which will return all books from all `shelfs` that meets the criteria of passed regex argument.

## Eight test

Create `User` class that will have:

- `id` parameter which will be unique for each `User`, and incremented every time that new `User` object will be created. This parameter must start from 0.

Add `users` parameter to `Library` class. It should be stored as `Set`, that there is no possibility to duplicate users in given library.

## Ninth test

Add `addUser` function to `Library` that will add given user to `users`

## Tenth test

Add boolean parameter `borrowed` to `Book`.

Add parameter `books` to `User`. It will store all books that was borrowed by user.

Add function `borrowABook` to `Library` which will get two parameters:

- `user` -> for which user the book will be borrowed
- `book` -> which book will be borrowed for given user

This function will:

- check if there is a given user in `users`
  - If there is not throw an `UndefinedUser` error
- if there is a given user, look for given book in all shelfs on library
  - if there is no such book throw and `NoSuchBookOnShelf` error
- if there is an user and there is search book find if this book isn't already borrowed
  - if is is borrowed throw an error `AlreadyBorrowed`
- if the book is not borrowed, add book to user books and set book as borrowed