/\*\*

\* The dataStore below will be used in the given console.log debug statements

\* at the bottom of the file.

\* You can modify this if you want to do your own testing.

\*

\* We will be using a modified dataStore in the automarking - see the

\* "Testing" section the specification.

\*/

const dataStore = {

academics: [

{

id: 10,

name: 'Ada',

hobby: 'music',

},

{

id: 20,

name: 'Ben',

hobby: 'gym',

},

{

id: 30,

name: 'Cid',

hobby: 'chess',

},

{

id: 40,

name: 'Dan',

hobby: 'art',

},

{

id: 50,

name: 'Eve',

hobby: 'yoga',

},

],

courses: [

{

id: 1511,

name: 'COMP1511',

description: 'Programming Fundamentals',

staffIds: [10, 20],

memberIds: [10, 20, 30, 40, 50],

},

{

id: 1521,

name: 'COMP1521',

description: 'Computer Systems Fundamentals',

staffIds: [20],

memberIds: [20, 40, 50],

},

{

id: 1531,

name: 'COMP1531',

description: 'Software Engineering Fundamentals',

staffIds: [20, 30],

memberIds: [20, 30, 10, 40],

},

],

};

/\*\*

\* @returns {{numAcademics: number}} object

\*/

function getNumAcademics() {

// TODO: Observe the return object, then replace with your implementation

// to work on dataStores with a different number of academics and courses.

return {

numAcademics: -1,

};

}

/\*\*

\* @returns {{numCourses: number}}

\*/

function getNumCourses() {

// TODO: Observe the return object, then replace with your implementation

// to work on dataStores with a different number of academics and courses.

return {

numCourses: -1,

};

}

/\*\*

\* @param {number} academicId - unique identifier for an academic.

\* @returns {{academic: {name: string, hobby: string}}}

\* @returns {{error: string}} on error

\*/

function getAcademicDetailsFromId(academicId) {

// TODO: Observe the return object, then replace with your implementation

// to work on dataStores with a different number of academics and courses.

return {

academic: {

name: 'Ada',

hobby: 'music',

},

};

}

/\*\*

\* @param {number} courseId - unique identifier for a course.

\* @returns {{course: {name: string, description: string}}}

\* @returns {{error: string}} on error

\*/

function getCourseDetailsFromId(courseId) {

// TODO: Observe the return object, then replace with your implementation

// to work on dataStores with a different number of academics and courses.

return {

course: {

name: 'COMP1511',

description: 'Programming Fundamentals',

},

};

}

/\*\*

\* @param {number} academicId - unique indentifier for an academic

\* @param {number} courseId - unique identifier for a course

\* @returns {{isMember: boolean}}

\* @returns {{error: string}} on error

\*/

function checkAcademicIsMember(academicId, courseId) {

// TODO: Observe the return object, then replace with your implementation

// to work on dataStores with a different number of academics and courses.

return {

isMember: false,

};

}

/\*\*

\* @param {number} academicId - unique indentifier for an academic

\* @param {number} courseId - unique identifier for a course

\* @returns {{isStaff: Boolean}}

\* @returns {{error: string}} on error

\*/

function checkAcademicIsStaff(academicId, courseId) {

// TODO: Observe the return object, then replace with your implementation

// to work on dataStores with a different number of academics and courses.

return {

isStaff: false,

};

}

console.log(`

\* You will not be able to compare two objects with '==='.

\* For week 1 and week 2, you can simply use console.log() and visually

\* compare the output line by line.

\*

\* NOTE: the output of any console.log statements, e.g. colours/whitespaces

\* does not matter when we mark your code, as we will be assessing the

\* returned objects from your functions directly.

\*

\* This means that if a number appears as brown/yellow, a string appears as

\* green, or some part of your output is on a different line, is is all okay!

\* This is simply how NodeJS format their output :).

\*/

`)

console.log('1. numAcademics()');

console.log('Expect:', { numAcademics: 5 });

console.log('Output:', getNumAcademics());

console.log();

console.log('2. numCourses()');

console.log('Expect:', { numCourses: 3 });

console.log('Output:', getNumCourses());

console.log();

console.log('3. getAcademicDetailsFromId(10)');

console.log('Expect:', { academic: { name: 'Ada', hobby: 'music' } });

console.log('Output:', getAcademicDetailsFromId(10));

console.log();

console.log('4. getAcademicDetailsFromId(999999)');

console.log('Expect:', { error: 'any relevant message (you should choose something meaningful!)' });

console.log('Output:', getAcademicDetailsFromId(999999));

console.log();

console.log('// TODO: You can add more debugging console.log here.');

Open the file [academics.js](file:////COMP1531/24T3/students/z5547184/lab01_academics/-/blob/master/academics.js) in your preferred text editor. The stub code (interface with a fake temporary implementation) for each function has been provided for you.

Before replacing them with your implementation, pay close attention to the returned object and see how it aligns with the [Interface: Functions](#interface-fuctions) and [Interface: Data Types](#interface-data-types).

Complete the functions getNumAcademics, getNumCourses, getAcademicDetailsFromId, getCourseDetailsFromId, checkAcademicIsMember, checkAcademicIsStaff.