**Задача: Управление библиотекой**

**Цель**: создать систему управления для небольшой библиотеки. Ваша задача - разработать функционал для учета книг и запросов читателей.

Базовые требования:

У вас есть массив объектов books, где каждый объект представляет книгу с полями: id (уникальный идентификатор), title (название), author (автор), и isAvailable (доступность книги для выдачи).

Есть массив объектов requests, где каждый объект содержит userId (идентификатор пользователя) и bookId (идентификатор запрашиваемой книги).

**Задачи**:

1. Проверка доступности книги: Напишите функцию, которая принимает bookId и возвращает true, если книга доступна для выдачи, и false - в противном случае.
2. Поиск книги по автору: Разработайте функцию, которая принимает имя автора и возвращает массив всех книг этого автора, доступных в библиотеке.
3. Обработка запросов на выдачу книг: Создайте функцию, которая обрабатывает массив requests, проверяя каждый запрос. Если книга доступна, функция должна обновить статус isAvailable на false и выводить сообщение о том, что книга выдана пользователю. Если книга недоступна, выводить сообщение о том, что книга уже выдана.
4. Возврат книги: Напишите функцию, которая принимает bookId и обновляет статус isAvailable данной книги на true, с выводом сообщения о возврате книги.

**Пример данных**:

const books = [

{ id: 1, title: "Война и мир", author: "Лев Толстой", isAvailable: true },

{ id: 2, title: "Гарри Поттер и философский камень", author: "Дж. К. Роулинг", isAvailable: false },

{ id: 3, title: "Улисс", author: "Джеймс Джойс", isAvailable: true }

];

const requests = [

{ userId: 101, bookId: 1 },

{ userId: 102, bookId: 2 },

{ userId: 103, bookId: 3 }

];

Результаты вызовов функций, которые вы напишете (для самопроверки). Можете использовать такие же названия функций

// Проверяем доступность книги с ID 1

console.log(isBookAvailable(books, 1)); // Должно вернуть true

// Находим книги автора "Лев Толстой"

console.log(findBooksByAuthor(books, "Лев Толстой")); // Должен вернуть массив с одной книгой

// Обрабатываем запросы на выдачу книг

processRequests(books, requests);

// Возвращаем книгу с ID 1

returnBook(books, 1);

// Проверяем, снова доступна ли книга с ID 1

console.log(isBookAvailable(books, 1)); // Должно вернуть true

**Требования к выполнению**:

Используйте циклы для обхода массивов.

Примените условные конструкции для проверки доступности книг.

Управляйте объектами для обновления их состояния.

**Задача: Система управления заказами в ресторане**

**Цель**: создать систему для управления заказами в ресторане. Ваша задача - разработать функционал для отслеживания заказов, учета запасов ингредиентов и обновления статуса заказов.

**Базовые требования**:

У вас есть массив объектов ingredients, где каждый объект представляет ингредиент с полями: id (уникальный идентификатор), name (название), и quantity (количество на складе).

Есть массив объектов dishes, где каждый объект содержит id (уникальный идентификатор), name (название блюда), и ingredients (массив идентификаторов ингредиентов, необходимых для приготовления).

Есть массив объектов orders, где каждый объект содержит orderId (уникальный идентификатор заказа), dishId (идентификатор заказанного блюда), и status (статус заказа, например, "новый", "в процессе", "готово", "отменено").

**Задачи**:

1. Проверка возможности приготовления блюда: Напишите функцию, которая принимает dishId и проверяет, достаточно ли ингредиентов на складе для приготовления этого блюда. Функция должна возвращать true, если блюдо можно приготовить, и false - в противном случае.
2. Обновление запасов ингредиентов: Разработайте функцию, которая обновляет количество ингредиентов на складе после приготовления блюда.
3. Изменение статуса заказа: Создайте функцию, которая изменяет статус заказа в массиве orders на основе переданного orderId и нового статуса.
4. Автоматическая обработка заказов: Напишите функцию, которая автоматически обрабатывает все новые заказы, проверяя возможность их приготовления и, в случае возможности, обновляя запасы и статус заказа.

**Пример данных**:

const ingredients = [

{ id: 1, name: "Томаты", quantity: 10 },

{ id: 2, name: "Моцарелла", quantity: 5 },

{ id: 3, name: "Базилик", quantity: 8 }

];

const dishes = [

{ id: 1, name: "Маргарита", ingredients: [1, 2, 3] }

];

const orders = [

{ orderId: 1, dishId: 1, status: "новый" }

];

**Требования к выполнению**:

Используйте циклы для обхода массивов.

Примените условные конструкции для проверки наличия ингредиентов и изменения статусов заказов.

Управляйте объектами для обновления их состояния в соответствии с логикой приложения.