const getIDs = () => {

return new Promise((resolve, reject) => {

setTimeout(() => {

resolve([532, 543, 753, 1, 5]);

}, 1500);

});

};

const getRecipe = (recipeID) => {

return new Promise((resolve, reject) => {

setTimeout((ID) => {

const recipe = {

title: "Fresh tomato pasta",

publisher: "Pinchas Hodadad",

};

resolve(`${ID}: ${recipe.title}`);

}, 1500, recipeID);

});

};

async function getRecipeById() {

try {

const IDs = await getIDs();

console.log(IDs);

const recipe = await getRecipe(IDs[2]);

console.log(recipe);

} catch (error) {

console.log("It is an error!");

}

}

getRecipeById();

In the modified code, we've defined an asynchronous function getRecipeById() that uses the await keyword to wait for the getIDs() and getRecipe() promises to resolve before moving on to the next step. The try-catch block is used to handle any errors that might occur during the execution of the promises. Finally, we call the getRecipeById() function to execute the code.