Q1: Which one would you prefer and why?
Both of the functions return the same result for a two dimensional integer array (1024x1024). Which one would you prefer? List pros and/or cons of both functions.

- sumB is better from a performance point of view as we are using loop-unrolling. It does manual loop unrolling which with the distributed sum variables works well with cpu pipelining.

- in sumA, loops are in order that subsequent adds jump in memory by SIZE*sizeof(int) bytes each, this is bad for cache

- sumA knows the exact dimensions of arrays and better in that sense since for sumB we need to be careful that whether the array a is a multiple of 4 or not and since we are not checking this, sumB could cause the program to crash.

- sumA is more typesafe

Q2: What controversies do you see in the following API example?
User object API doc

It should be as follows: put method should be used for update/rename if we are sending the whole user data, we also can use path method for update/modify the id and time-zone of users with just sending id and time zone data not the whole data.

GET /users
POST /users/new
PUT or PATCH /users/:id/update (perform /update id)
PUT or PATCH /users/:id/rename (still just an update)
PUTor PATCH /users/:id/update-timezone (still another type of update)
DELETE /users/delete ? id

Q3: What problems related to password security can you see in the following example?

- Using salted md5 for passwords is a bad idea. Not because of MD5's cryptographic weaknesses, but because it's fast. This means that an attacker can try billions of candidate passwords per second on a single GPU.

- WE should use a new random salt per password

- We should not try to implement secure password storage on our own but use existing (verified and secure) solutions like Rails' builtin has_secure_password option.

Q4: What would you give as a feedback for a pull request including this code?

Better to define variables as follows:
const deposits = user.transactions.history.deposits;
let sum = 0;

- better to use let keyword as opposed to var keyword since is an improved version of the *var* keyword in js.

- let keyword allows you to declare variables that are limited in scope to the block, statement, or expression on which it is used. This is unlike the var keyword, which defines a variable globally, or locally to an entire function regardless of block scope.

- use const keyword for the variable deposits since we do not want to update it along the code.

Q5: Find database related issues

This part could be wrong:
db.game_accounts.update({name: from}, {$set: {credits: fromAccount.credits - amt}});
db.game_accounts.update({name: to}, {$set: {credits: toAccount.credits + amt}});

- This is because there might be another transaction happening between the findOne() and the update(). You're better using $inc to increment the value of a field by a specified amount which will perform the same operation.

- MongoDB is not really meant for this kind of transaction, you might prefer using another kind of DB (such as **SQL database**) which can make sure that the data is up to date and all operations are tracked and done. A SQL database is a great fit for transaction-oriented systems such as customer relationship management tools.

Q6: What problems does this method have?

- Based on conventional naming in javascript, we need use small letters for method variables (isCredit should be iscredit) because class method names are written as isCredit with capital letter "C". also.

- Might pass the default value for iscredit or write a separate method to generate True/false val for credit and call that method in if condition here.

Q7: List four typical solutions to optimize database

- Optimize Queries
- Create optimal indexes
- Get a stronger CPU
- Allocate more memory